

# Building a Good Team: Secretary Problems and the Supermodular Degree

Moran Feldman\*

Rani Izsak†

## Abstract

In the (classical) SECRETARY PROBLEM, one has to hire the best among  $n$  candidates. The candidates are interviewed, one at a time, at a uniformly random order, and one has to decide on the spot, whether to hire a candidate or continue interviewing. It is well known that the best candidate can be hired with a probability of  $1/e$  (Dynkin, 1963). Recent works extend this problem to settings in which multiple candidates can be hired, subject to some constraint. Here, one wishes to hire a set of candidates maximizing a given objective set function.

Almost all extensions considered in the literature assume the objective set function is either linear or submodular. Unfortunately, real world functions might not have either of these properties. Consider, for example, a scenario where one hires researchers for a project. Indeed, it can be that some researchers can substitute others for that matter. However, it can also be that some combinations of researchers result in synergy (see, *e.g.*, Woolley et al., Science 2010, for a research about collective intelligence). The first phenomenon can be modeled by a submodular set function, while the latter cannot.

In this work, we study the secretary problem with an *arbitrary* non-negative monotone valuation function, subject to a general matroid constraint. It is not difficult to prove that, generally, only very poor results can be obtained for this class of objective functions. We tackle this hardness by combining the following: (1) Parametrizing our algorithms by the *supermodular degree* of the objective function (defined by Feige and Izsak, ITCS 2013), which, roughly speaking, measures the distance of a function from being submodular. (2) Suggesting an (arguably) natural model that permits approximation guarantees that are *polynomial* in the supermodular degree (as opposed to the standard model which allows only *exponential* guarantees). Our algorithms learn the input by running a non-trivial estimation algorithm on a portion of it whose size depends on the supermodular degree.

We also provide better approximation guarantees for the special case of a uniform matroid constraint. To the best of our knowledge, our results represent the first algorithms for a secretary problem handling arbitrary non-negative monotone valuation functions.

---

\*School of Computer and Communication Sciences, EPFL, Switzerland. Email: [moran.feldman@epfl.ch](mailto:moran.feldman@epfl.ch).

†Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Israel. Email: [ran.izsak@weizmann.ac.il](mailto:ran.izsak@weizmann.ac.il).

# 1 Introduction

In the (classical) SECRETARY PROBLEM, one has to hire a worker from a pool of  $n$  candidates. The candidates arrive to an interview at a uniformly random order, and the algorithm must decide immediately and irrevocably, after interviewing a candidate, whether to hire him or continue interviewing. The objective is to hire the best candidate. It is well-known that the best candidate can be hired with a probability of  $1/e$ , and that this is asymptotically optimal [11].

Recently, there has been an increased interest in variants of the secretary problem where more than a single candidate can be selected, subject to some constraint (*e.g.*, a matroid constraint). Such variants have important applications in mechanism design (see, *e.g.*, [2, 3, 4, 24] and the references therein). When more than one candidate can be selected, there is a meaning to the values of *subsets* of candidates. If one allows these values to be determined by an arbitrary non-negative monotone set function, then only exponentially competitive ratios (in the number of candidates) can be achieved, even subject to a simple cardinality constraint.<sup>1</sup>

In light of the above hardness, previous works have concentrated on restricted families of objective functions, such as linear and submodular functions (see, *e.g.*, [5, 6, 8, 9, 17, 20, 26] and a more thorough discussion in Section 3.2). However, for many applications, the desired set function might admit complements, *i.e.*, a group of candidates might exhibit synergy and contribute more as a group than the sum of the candidates' personal contributions (see also Woolley et al. [29] for a research about collective intelligence). Such complements cannot be modeled by submodular (or linear) objectives. Dealing with complements in general results in unacceptable guarantees, as discussed above. However, what if one has a function which is submodular, except for a pair of candidates which are better to hire together? Can we guarantee anything for this case?

In this paper, we give a strong affirmative answer to this question. Specifically, we give algorithms for secretary problems with *arbitrary* non-negative monotone objective functions, whose guarantees are proportional to the distance of the objective function from being submodular, as measured by the *supermodular degree* (defined by [14]). Back to the above example, the pair of synergistic candidates results in an objective function with a supermodular degree of 1, and for such an objective our algorithms provide a constant competitive ratio for the problem of hiring a team of a given size. Intuitively, the supermodular degree can be seen as measuring the number of candidates that any single candidate can have synergy with. Our algorithms handle both the case of a cardinality constraint (demonstrated above) and the more general case of a matroid constraint. For a cardinality constraint, we obtain a constant competitive ratio when the supermodular degree is constant. For a (general) matroid constraint, our competitive ratios depend logarithmically on the rank of the matroid.<sup>2</sup> To the best of our knowledge, these are the first algorithms for the secretary problem with an arbitrary non-negative monotone objective set function.

## 2 Preliminaries

For completeness of the presentation, we give in this section a few relevant definitions from the literature (see, *e.g.*, [15]). All set functions in this work are non-negative and (non-decreasing) monotone<sup>3</sup>. For readability, given a set  $S \subseteq \mathcal{N}$  and an element  $u \in \mathcal{N}$  we use  $S + u$  to denote  $S \cup \{u\}$  and  $S - u$  to denote  $S \setminus \{u\}$ .

---

<sup>1</sup>Intuitively, the bad example consists of a cardinality constraint allowing us to select only  $k$  candidates, and an objective function assigning a strictly positive value only to sets containing  $k$  specific candidates or more than  $k$  candidates. In this case the algorithm has no room for mistakes, which leads to a very poor performance.

<sup>2</sup>Note that, till a very short while ago, this was the case even for the state of the art algorithm for a *submodular* objective function (which corresponds to a supermodular degree of 0) [20]. An improved algorithm with a competitive ratio of  $O(\log \log k)$  (where  $k$  is the rank of the matroid) has been recently given by [18].

<sup>3</sup>A set function  $f: 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$  is monotone if and only if  $f(S) \leq f(T)$  whenever  $S \subseteq T \subseteq \mathcal{N}$ .

## 2.1 Matroids

Given a ground set  $\mathcal{N}$ , a pair  $(\mathcal{N}, \mathcal{I})$  is called a **matroid** if  $\mathcal{I} \subseteq 2^{\mathcal{N}}$  obeys three properties:

- (i)  $\mathcal{I}$  is non-empty.
- (ii)  $\mathcal{I}$  is hereditary, *i.e.*,  $S \subseteq T \subseteq \mathcal{N}$  and  $T \in \mathcal{I}$  imply  $S \in \mathcal{I}$ .
- (iii) For every two sets  $S, T \in \mathcal{I}$  such that  $|S| > |T|$ , there exists an element  $u \in S \setminus T$ , such that  $T + u \in \mathcal{I}$ . This property is called the *augmentation property* of matroids.

We say that a set  $S \subseteq \mathcal{N}$  is *independent* if  $S \in \mathcal{I}$ . The rank of a matroid is the size of the largest independent set in  $\mathcal{I}$ . One important class of matroids, which is central to our work, is the class of *uniform* matroids. A uniform matroid of rank  $k$  is simply a cardinality constraint, *i.e.*, a set  $S \subseteq \mathcal{N}$  is independent in such a matroid if and only if its cardinality is at most  $k$ .

## 2.2 Supermodular degree

The following standard definition is very handy.

**Definition 2.1** (Marginal set function). *Let  $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$  be a set function and let  $u \in \mathcal{N}$ . The **marginal set function** of  $f$  with respect to  $u$ , denoted by  $f(u \mid \cdot)$  is defined as  $f(u \mid S) \stackrel{\text{def}}{=} f(S + u) - f(S)$ . When the underlying set function  $f$  is clear from the context, we sometimes call  $f(u \mid S)$  the **marginal contribution** of  $u$  to the set  $S$ . Similarly, for subsets  $S, T \subseteq \mathcal{N}$ , we use the notation  $f(T \mid S) \stackrel{\text{def}}{=} f(S \cup T) - f(S)$ .*

We can now give the definition of the supermodular degree (originally defined by [14]), which is used to parameterize our results.

**Definition 2.2** (Supermodular (dependency) degree). *The **supermodular degree** of an element  $u \in \mathcal{N}$  with respect to  $f$  is defined as the cardinality of the set  $\mathcal{D}_f^+(u) = \{v \in \mathcal{N} \mid \exists S \subseteq \mathcal{N} f(u \mid S + v) > f(u \mid S)\}$ , containing all elements whose existence in a set might increase the marginal contribution of  $u$ .  $\mathcal{D}_f^+(u)$  is called the **supermodular dependency set** of  $u$  with respect to  $f$ , and we sometimes refer to the elements of  $\mathcal{D}_f^+(u)$  as **supermodular dependencies**. The **supermodular degree** of a function  $f$ , denoted by  $\mathcal{D}_f^+$ , is simply the maximum supermodular degree of any element  $u \in \mathcal{N}$ . Formally,  $\mathcal{D}_f^+ = \max_{u \in \mathcal{N}} |\mathcal{D}_f^+(u)|$ . When the underlying set function is clear from the context, we sometimes omit it from the notations.*

Note that  $0 \leq \mathcal{D}_f^+ \leq n - 1$  for any set function  $f$ . More specifically,  $\mathcal{D}_f^+ = 0$  when  $f$  is *submodular*, and becomes larger as  $f$  deviates from submodularity. When the function  $f$  is clear from the context, we use  $d$  to denote  $\mathcal{D}_f^+$ .

## 2.3 Input representation

In general, a set function might assign  $2^n$  different values for the subsets of a ground set of size  $n$ . Thus, not every set function has a succinct (*i.e.*, polynomial in  $n$ ) representation. Therefore, it is a common practice to assume access to a set function via an oracle. That is, an algorithm handling a set function often gets access to an oracle that answers queries about the function, instead of getting an explicit representation of the function. Arguably, the most basic type of an oracle is the *value oracle*, which, given any subset of the ground set, returns the value assigned to it by the set function. Formally:

**Definition 2.3.** *Value oracle of a set function  $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$  is the following:*

*Input:* A subset  $S \subseteq \mathcal{N}$ .

*Output:*  $f(S)$ .

Similarly, since in a given matroid the number of independent subsets might be, in general, exponential in the size of the ground set, it is common to assume access to the following type of oracle.

**Definition 2.4.** *Independence oracle of a matroid  $(\mathcal{N}, \mathcal{I})$  is the following:*

*Input:* A subset  $S \subseteq \mathcal{N}$ .

*Output:* A Boolean value indicating whether  $S \in \mathcal{I}$ .

Additionally, in order to manipulate a function with respect to the supermodular degree, one needs a way to determine the supermodular dependencies of a given element of the ground set. An oracle for that purpose was introduced by [14], and was later used also by [15]. Formally:

**Definition 2.5.** *Supermodular oracle of a set function  $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$  is the following:*

*Input:* An element  $u \in \mathcal{N}$ .

*Output:* The set  $\mathcal{D}^+(u)$  of the supermodular dependencies of  $u$  with respect to  $f$ .

The above oracles definitions are acceptable for offline algorithms. In online settings, these oracles have to be weakened and limited to return only information that we “expect” the algorithm to have. Some possible weakened versions can be found in previous work. Still, finding a set of weakened oracles that “makes sense” in the context of the supermodular degree is not trivial. Our model, including the weakened oracles that we use, appears in Section 3.

## 2.4 Online algorithms

Like standard *online* algorithms, the performance of a secretary algorithm is measured by the **competitive ratio**, which is the worst case ratio between the expected performance of the algorithm and the performance of an offline optimal algorithm. More formally, let  $\mathcal{P}$  be the set of possible instances,  $OPT(P)$  be the value of the optimal solution for an instance  $P \in \mathcal{P}$  and  $ALG(P)$  be the value of the algorithm’s solution given the instance  $P$ . Then, the competitive ratio (for maximization problems) of the algorithm is given by:

$$\sup_{P \in \mathcal{P}} \frac{OPT(P)}{\mathbb{E}[ALG(P)]} ,$$

where the expectation is over the randomness of the algorithm and the arrival order of the input.

## 2.5 Techniques

Most algorithms for secretary problems start with a learning phase in which they reject all elements, and later, after accumulating some information about the input, they move to a phase in which they may accept elements. When the value of an element might positively depend on other  $d$  elements, there might be a set of  $d + 1$  elements such that every reasonable solution must contain this set. In this case, any reasonably good algorithm must terminate the learning phase, with a significant probability, before any element of this set arrives.

This means that the learning phase of our algorithms consists of only about  $1/d$  of the input (where  $d$  is the supermodular degree of the objective function), and thus, they rarely see all the dependencies of an element in the learning phase. Hence, by the end of the learning phase, our algorithms cannot calculate an optimal solution for the sub-problem represented by the part of the input seen thus far. However, we show that it is possible to *estimate* the value of the optimal solution based on the learning phase, and this estimation is crucial for the performance of our algorithms.

### 3 Model and results

Consider the following scenario. A client enters a store, and wants to buy herself a new phone. However, the client's main motive to buy this phone is a novel accessory which is not supported by her old phone. Thus, the client asks the salesman to buy the phone bundled with the accessory. Unfortunately, the accessory is not available at that time, because supply does not meet the overwhelming demand. If the client insists on buying the phone only bundled with the accessory, then the salesman can offer her to buy a phone now and get the accessory next week when a new supply shipment arrives.

On the other hand, consider a slightly different scenario. In this scenario, the client tells the salesman that she wants the phone together with some accessory, but does not tell him which accessory it is. The client then offers the following deal: the salesman will give her the phone now, and the client will pay when the unspecified accessory becomes available. Clearly no salesman can accept such an offer.

Our model (below) assumes the more realistic first scenario. That is, in case of complementarity, the "bidder" is required to announce the future elements she needs in order to get the maximum value from the current "product".

Formally, an instance of the monotone matroid secretary problem consists of a ground set  $\mathcal{N}$  of size  $n$ , a non-negative monotone set function  $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$  and a matroid  $M = (\mathcal{N}, \mathcal{I})$ . The execution of an algorithm for this problem consists of  $n$  steps (also referred to as times). In each step the following occurs:

- One element of  $\mathcal{N}$  is revealed (arrives), at a uniformly random order (without repetitions).
- The algorithm must decide whether to include the element in its output (irrevocably).

The objective of the algorithm is to select an independent subset maximizing  $f$ . When making decisions, the algorithm has access to the value of  $n$ , the supermodular degree of  $f$  and the following oracles. The first oracle is the independence oracle given above, which gives information about the constraint. The second oracle gives information about the objective function. This oracle is the counterpart of the value oracle defined above.

**Definition 3.1.** *Online marginal oracle of a set function  $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$  is the following:*

*Input:* An element  $u \in \mathcal{N}$  that has already been revealed and a subset  $S \subseteq \mathcal{N}$ .

*Output:*  $f(u \mid S)$ .

Note that  $S$  does not have to be fully (or even partially) revealed, but  $u$  does have to. That is, we can only ask marginal queries for elements that have been revealed, but we can ask for their marginal value with respect to any subset. Our algorithms use the online marginal oracle only for the purpose of finding the best marginal that an element  $u$  can have with respect to already accepted elements and subsets of  $\mathcal{D}^+(u)$ . This use is consistent with the above motivation of our model.

The last oracle of our model returns the dependency sets of already revealed elements.

**Definition 3.2.** *Online supermodular oracle of a set function  $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$  is the following:*

*Input:* An element  $u \in \mathcal{N}$  that has already been revealed.

*Output:*  $\mathcal{D}_f^+(u)$ .

Observe that the last oracle can, in fact, be implemented using the online marginal oracle, albeit using an exponential time complexity in  $n$ . However, this oracle is a natural online variant of the supermodular oracle, and thus, it emphasizes the relation between our model and previous work on the supermodular oracle. If time complexity is not a priority, as is often the case when analyzing online algorithms, then every use of this oracle can be replaced by an appropriate procedure using the online marginal oracle.

**Additional Notation.** In the rest of this paper we use the notation  $\mathcal{N}_u$  to denote the set of elements revealed up to the point in which a given element  $u \in \mathcal{N}$  is revealed (*i.e.*,  $\mathcal{N}_u$  contains  $u$  and every other element revealed before  $u$ ).

### 3.1 Our results

In this section we formally state our results for the model introduced above. Our first result is for instances where the matroid  $M$  is of rank  $k \leq \mathcal{D}_f^+ + 1$ . This setting is interesting for two reasons: it is closely related to the classical secretary problem, and our algorithm for it is used as a building block in our algorithms for other settings.

**Theorem 3.1.** *There exists an  $O(k \log k) = O(\mathcal{D}_f^+ \log \mathcal{D}_f^+)$ -competitive algorithm for the monotone matroid secretary problem when the rank of the matroid constraint  $M$  is  $k \leq \mathcal{D}_f^+ + 1$ .*

The time complexity of the above algorithm (and all our other algorithms) is  $\text{Poly}(n, 2^{\mathcal{D}_f^+})$ . The exponential dependence of the time complexity on  $\mathcal{D}_f^+$  is unavoidable even for *offline* algorithms and a uniform matroid constraint (see [15] for more details). Our main result is given by the next theorem.

**Theorem 3.2.** *There exists an  $O(\mathcal{D}_f^{+3} \log \mathcal{D}_f^+ + \mathcal{D}_f^{+2} \log k)$ -competitive algorithm for the monotone matroid secretary problem.*

Interestingly, the above competitive ratio matches the, till recently, state of the art ratio of  $O(\log k)$  by Gupta et al. [20] for the case where  $f$  is a submodular function, and extends it to any constant supermodular degree. For uniform matroids we have the following improved guarantee.

**Theorem 3.3.** *There exists an  $O(\mathcal{D}_f^{+3} \log \mathcal{D}_f^+)$ -competitive algorithm for the monotone matroid secretary problem when the matroid  $M$  is uniform.*

Note that the guarantee of Theorem 3.3 has no dependence on  $k$ , and thus, it yields a constant competitive ratio for a constant supermodular degree.

It is handy to assume that  $f$  is normalized (*i.e.*,  $f(\emptyset) = 0$ ). Reduction 1 in Appendix A shows that this assumption is without loss of generality, and thus, we implicitly assume it in all our proofs.

**Lower Bounds.** Note that even the *offline* version of maximizing a function  $f$  with respect to matroid constraint is  $\mathcal{NP}$ -hard to approximate within a guarantee of  $\Omega(\ln \mathcal{D}_f^+ / \mathcal{D}_f^+)$  (see, *e.g.*, [15, 21]). Moreover, for a uniform matroid constraint, it is *SSE*-hard to achieve any constant<sup>4</sup> approximation guarantee (even) in the offline setting [15].

### 3.2 Related results

**Secretary problem.** Many variants of the secretary problem have been considered throughout the years, and we mention here only those most relevant to this work. Under a cardinality constraint of  $k$ , Babaioff, Immorlica, Kempe and Kleinberg [3] and Kleinberg [24] achieve two incomparable competitive ratios of  $e$  and  $1/[1 - O(1/\sqrt{k})]$ , respectively, for linear objective functions. For submodular objective functions, the best algorithms have a competitive ratio of  $8e^2 \approx 59$  for the general case [6] and a competitive ratio of  $(e^2 + e)/(e - 1) \approx 5.88$  when the objective is also monotone [16].

---

<sup>4</sup>That is, a guarantee that does not depend on  $\mathcal{D}_f^+$ .

The matroid secretary problem considers a linear objective and a general matroid constraint. This variant was introduced by Babaioff, Immorlica and Kleinberg [5], who described an  $O(\log k)$ -competitive algorithm for it (where  $k$  is the rank of the matroid) and conjectured the existence of an  $O(1)$ -competitive algorithm. Motivated by this conjecture,  $O(1)$ -competitive algorithms have been obtained for a wide variety of special classes of matroids including graphic matroids [5, 25], transversal matroids [5, 9, 25], co-graphic matroids [28], linear matroids with at most  $k$  non-zero entries per column [28], laminar matroids [22, 23, 27], regular matroids [10], and some types of decomposable matroids, including max-flow min-cut matroids [10]. However, progress on the general case has been much slower. An  $O(\sqrt{\log k})$ -competitive algorithm was described by Chakraborty and Lachish [8], and very recently two  $O(\log \log k)$ -competitive algorithms were given by Lachish [26] and Feldman, Svensson and Zenklusen [17].

The submodular variant of the matroid secretary problem was also considered. For general matroids [20] gave an  $O(\log k)$ -competitive algorithm, and  $O(1)$ -competitive algorithms were described for special classes of matroids including partition matroids [6, 16, 20] and transversal and laminar matroids [27]. A recent work [18] shows that any algorithm for the linear variant can be translated, with a limited loss in the competitive ratio, into an algorithm for the submodular variant. This implies an  $O(1)$ -competitive algorithm for the submodular variant under any class of matroids admitting such an algorithm for linear objectives, and an  $O(\log \log k)$ -competitive algorithm for general matroids. A secretary problem with an even more general family of objective functions was considered by Bateni, Hajiaghayi and Zadimoghaddam [6] who proved a hardness result for the family of subadditive objective functions. Finally, variants of the matroid secretary problem which use a different arrival process or a non-adversarial assignment of element values were also considered [19, 23, 28].

**Complexity measures of set functions.** Complexity measures of set functions have been previously studied. Abraham, Babaioff, Dughmi and Roughgarden [1] studied the welfare maximization problem with respect to a complexity measure that gives greater values to set functions (*i.e.*, mark them as more “complex”) as their complementarity increases, in some sense. This complexity measure is applicable only to a restricted class of set functions. The notion of supermodular degree, which we use in this work, was introduced by [14], again, with an application to the welfare maximization problem, and was later used for a more general application by [15]. A stronger complexity measure for set functions was studied by [13]. However, their results assume access to a demand oracle (see Blumrosen and Nisan [7]), which is common in the context of combinatorial auctions, but, to the best of our knowledge, has not been used outside this world.

**Complements in online settings.** A different online setting exhibiting complements can be found in the work of Emek, Halldórsson, Mansour, Patt-Shamir, Radhakrishnan and Rawitz on online set packing [12].

## 4 Small rank matroids (Theorem 3.1)

In this section, we describe the main intuitive ideas behind the proof of Theorem 3.1. The proof itself is deferred to Appendix B. For simplicity, we assume in this section that  $M$  is a uniform matroid of rank  $d + 1$ . The extension to general matroids and smaller ranks is quite straightforward.

A natural generalization of the algorithm for the classical secretary problem is the following algorithm. First, during the learning phase, reject the first  $O(n/d)$  elements. From the remaining elements, take the first one whose marginal contribution with respect to some of its future

supermodular dependencies (*i.e.*, the elements that may increase its marginal contribution and the algorithm can still choose to take) is better than any such contribution inspected thus far.

It is not difficult to argue that the best marginal contribution seen by the above algorithm is always at least  $f(OPT)/(d+1)$ . However, to get a competitive ratio guarantee, we need to show that the algorithm manages to pick this best contribution with a significant probability. One approach to proving this claim is by generalizing the analysis of the classical secretary algorithm to this more general algorithm. Such a generalization requires lower bounding the probability that the following two events occur (at the same time).

- The element with best marginal contribution arrives after the learning phase.
- The second best marginal contribution, up to the point where we see the best contribution, is seen during the learning phase.

Unfortunately, it is difficult to bound the above probability due to the following phenomenon. The earlier an element arrives, the more future supermodular dependencies it has, and thus, the higher its corresponding marginal contribution. Hence, elements in the learning phase tend to have larger marginal contributions in comparison to elements appearing after the learning phase.

To overcome this issue, we modify the algorithm. Specifically, instead of comparing the marginal contribution of the current element to the marginal contributions seen thus far, we compare it to the marginal contributions that the elements seen thus far could have if they would have arrived at this time (instead of the time in which they have really arrived). A similar idea has been previously used by a work on the case of a submodular objective function [16].

Additionally, to get the exact approximation ratio guaranteed by Theorem 3.1, the algorithm has to use a random threshold from a logarithmic scale. This allows the analysis to assume that (with a significant probability) the learning phase takes about half of the time up to the point when the best contribution is observed by the algorithm.

## 5 Estimation aided algorithms

We say that a value  $\text{opt}_\alpha$  is an  $\alpha$ -estimation of an optimum solution  $OPT$  if it obeys  $f(OPT)/\alpha \leq \text{opt}_\alpha \leq f(OPT)$ . We say that an algorithm is  $\alpha$ -aided if it assumes getting an  $\alpha$ -estimation of the optimum as part of its input. In this section we describe an aided algorithm for the case of a general matroid constraint. An improved aided algorithm for the special case of a uniform matroid constraint can be found in Appendix D. In the next section we explain how to convert our aided algorithms into non-aided ones. Note that our aided algorithms work even under a model where the arrival order is determined by an adversary. However, the randomness of the input is required for converting them into non-aided algorithms.

**Theorem 5.1.** *For every  $\alpha \geq 1$ , there exists an  $\alpha$ -aided  $O(d^2 \log(\alpha k))$ -competitive algorithm for the monotone matroid secretary problem.*

The algorithm we use to prove Theorem 5.1 is Algorithm 1. A few of the ideas we use in Algorithm 1 and its analysis can be traced back to [5].

We define a weight  $w(u)$  for every element  $u \in OPT$  as follows:  $w(u) = f(u \mid OPT \setminus \mathcal{N}_u)$ . For ease of notation, we extend  $w$  to subsets of  $OPT$  in the natural way. Let us denote  $p_1 = -\lceil \log_2 k \rceil$  and  $p_2 = \lceil \log_2 \alpha \rceil$ . For every integer  $p_1 \leq p \leq p_2$ , we define a set (bucket)  $OPT_p = \{u \in OPT \mid 2^p \cdot \frac{\text{opt}_\alpha}{2} \leq w(u) \leq 2^p \cdot \text{opt}_\alpha\}$ . Intuitively speaking, the following lemma shows that there is sufficient value in all the buckets together. The lemma holds since every element that does not get into any bucket must have a very low weight.

**Lemma 5.2.**  $w\left(\bigcup_{p=p_1}^{p_2} OPT_p\right) \geq \frac{f(OPT)}{2}$ .



---

**Algorithm 1:**  $\alpha$ -Aided Algorithm for General Matroids

---

```

1 Let  $p$  be a uniformly random integer from the set  $\{-\lceil \log_2 k \rceil - 3, -\lceil \log_2 k \rceil - 2, \dots, \lceil \log_2 \alpha \rceil\}$ .
2 Let  $\tau \leftarrow 2^p \cdot \frac{\text{opt}_\alpha}{2}$ .
3 Let  $S \leftarrow \emptyset$ .
4 for every arriving element  $u$  do
5   if there exists a set  $D^*(u) \subseteq \mathcal{D}^+(u) \setminus \mathcal{N}_u$  such that  $f(u \mid D^*(u) \cup S) \geq \tau$  and
6      $S \cup D^*(u) + u \in \mathcal{I}$  then
7      $\quad$  └ Add  $D^*(u) + u$  to  $S$ .
8 return  $S$ .
```

---

We defer the proof of Lemma 5.2 and the other lemmata of this section to Appendix C. Our next objective is to show that if Algorithm 1 selects a value  $p$ , then its gain is proportional to  $w(OPT_{p+3})$ . Whenever  $S$  appears below it denotes the output of the algorithm.

**Lemma 5.3.** *If Algorithm 1 selects a value  $p$  and  $|S| \geq |OPT_{p+3}|/[2(d+1)]$ , then  $f(S) \geq w(OPT_{p+3})/[32(d+1)^2]$ .*

Intuitively, the last lemma holds since a large  $|S|$  means that the algorithm adds elements to  $S$  in many iterations, and each iteration increases  $f(S)$  by at least  $\tau$ .

**Lemma 5.4.** *If Algorithm 1 selects a value  $p$  and  $|S| < |OPT_{p+3}|/[2(d+1)]$ , then  $f(S) \geq w(OPT_{p+3})/8$ .*

The main idea behind the proof of Lemma 5.4 is as follows. Since  $|S|$  is small, many elements of  $OPT_{p+3} \setminus S$  could be added to it, together with their dependencies, without violating independence. The reason these elements were not added must be that they did not pass the threshold, which can only happen when  $f(S)$  is large enough.

**Corollary 5.5.** *If Algorithm 1 selects a value  $p$ , then  $f(S) \geq w(OPT_{p+3})/[32(d+1)^2]$ .*

We are now ready to prove Theorem 5.1.

*Proof of Theorem 5.1.* Recall that every value  $p$  is selected by Algorithm 1 with probability at least  $(\log_2 \alpha + \log_2 k + 6)^{-1} = (\log_2(\alpha k) + 6)^{-1}$ . Hence, by Corollary 5.5, the expected value of the output of Algorithm 1 is at least:

$$\frac{1}{\log_2(\alpha k) + 6} \cdot \sum_{p=p_1}^{p_2} \frac{w(OPT_{p+3})}{32(d+1)^2} = \frac{w\left(\bigcup_{p=p_1}^{p_2} OPT_p\right)}{32(d+1)^2 \cdot [\log_2(\alpha k) + 6]} \geq \frac{f(OPT)}{64(d+1)^2 \cdot [\log_2(\alpha k) + 6]},$$

where the last inequality is due to Lemma 5.2. □

## 6 Estimating the optimum: from aided to non-aided algorithms

In this section, we show how to convert aided algorithms into non-aided ones. Together with our aided algorithms, the following theorem implies the results stated in Theorems 3.2 and 3.3.

**Theorem 6.1.** *If there exists a  $(80(d+2)^2)$ -aided  $\beta$ -competitive algorithm  $ALG$  for the monotone matroid secretary problem with supermodular degree  $d$ , under a class  $\mathcal{C}$  of matroid constraints closed under restriction, then there also exists a non-aided  $O(d^3 \log d + \beta)$ -competitive algorithm for the same problem.*

Recall that the truncation of a matroid  $M = (\mathcal{N}, \mathcal{I})$  to rank  $k'$  is a matroid  $M' = (\mathcal{N}, \mathcal{I}')$  where a set  $S \subseteq \mathcal{N}$  is independent in  $M'$  if and only if  $S \in \mathcal{I}$  and  $|S| \leq k'$ . The algorithm we use to prove Theorem 6.1 is Algorithm 2.

---

**Algorithm 2:** Multiple Elements Estimation

---

```

1 with probability  $1/2$  do
2   Apply the algorithm guaranteed by Theorem 3.1 to the problem after truncating the
   matroid to rank  $\min\{k, d+1\}$  (where  $k$  is the rank of the original matroid).
3 otherwise
4   Choose  $X$  according to the binomial distribution  $B(n, (d+2)^{-1})$ , and let  $T$  the set of the
   first  $X$  elements revealed.
5   Let  $A \leftarrow \emptyset$  and  $W \leftarrow 0$ .
6   while there exist  $u \in T \setminus A$  and  $\mathcal{D}_u \subseteq \mathcal{D}^+(u)$  s.t.  $A \cup \mathcal{D}_u + u \in \mathcal{I}$  do
7     Find such a pair maximizing  $f(u \mid A \cup \mathcal{D}_u)$ .
8     Increase  $W \leftarrow W + f(u \mid A \cup \mathcal{D}_u)$ .
9     Update  $A \leftarrow A \cup \mathcal{D}_u + u$ .
10  Apply ALG to the remaining elements with  $\text{opt}_{80(d+2)^2} = W/10$ .
```

---

Algorithm 2 consists of two parts, each executed with probability  $1/2$ . In order to prove Theorem 6.1 we show that for any instance of the monotone matroid secretary problem, one of the following cases is true:

- The algorithm guaranteed by Theorem 3.1 produces an  $O(d^3 \log d)$ -competitive solution for the non-truncated problem.
- The second part of Algorithm 2 is  $O(\beta)$ -competitive.

To determine which of the above cases is true for every given instance we need some notation. Let  $u^* \in \mathcal{N}$  be an element maximizing

$$\max_{\substack{S \subseteq \mathcal{D}^+(u^*) \\ S + u^* \in \mathcal{I}}} f(u^* \mid S) ,$$

and let  $m^*$  and  $S^*$  denote the value of this maximum and an arbitrary corresponding set  $S$ , respectively. It can be shown quite easily that the first case above holds when  $m^* \geq f(OPT)/(256(d+1)^2)$  (see Lemma C.1 for details). Thus, we sketch here only the more interesting part of the analysis, which is to show that the second above case holds when  $m^* \leq f(OPT)/(256(d+1)^2)$  (a full proof can be found in Appendix E). The main thing that we need to show is that  $\text{opt}_{80(d+2)^2}$  is, with constant probability, a  $80(d+1)^2$ -estimation for  $f(OPT)$ . For that purpose, we relate the expected value of the estimate  $\text{opt}_{80(d+2)^2}$  to  $f(OPT)$ , and then bound the variance of this estimate to show that it is close enough to its expected value with constant probability.

In the rest of this section we assume Algorithm 2 executes its second part. Let  $W_\ell$  and  $A_\ell$  be  $W$  and  $A$ , respectively, when Algorithm 2 exits its loop. We bound  $W_\ell$ , which immediately implies bounds for  $\text{opt}_{80(d+2)^2}$ . In order to achieve that, we switch our attention from Algorithm 2 to an offline algorithm (Algorithm 3) producing exactly the same distribution for  $W_\ell$ . We explain intuitively why the distributions of  $W_\ell$  in both algorithms are the same. Let us choose a set  $T_{\text{pre}}$  ahead, exactly as  $T$  is chosen by the online algorithm. Next, we modify the offline algorithm so that whenever it chooses an element from its set  $T$ , instead of randomly deciding whether to keep it in  $T$ , it queries membership in  $T_{\text{pre}}$ . Clearly, since there are no repetitions in these queries, this does not affect the behavior of the offline algorithm. However, one can verify that the output of the offline algorithm is now identical to the output of the online algorithm when it selects  $T = T_{\text{pre}}$ .

---

**Algorithm 3:** Offline  $W$  Calculation

---

```

1 Let  $A \leftarrow \emptyset$ ,  $W \leftarrow 0$  and  $T \leftarrow \mathcal{N}$ .
2 while there exist  $u \in T \setminus A$  and  $\mathcal{D}_u \subseteq \mathcal{D}^+(u)$  s.t.  $A \cup \mathcal{D}_u + u \in \mathcal{I}$  do
3   Find such a pair maximizing  $f(u \mid A \cup \mathcal{D}_u)$ .
4   with probability  $(d+2)^{-1}$  do
5     Increase  $W \leftarrow W + f(u \mid A \cup \mathcal{D}_u)$ .
6     Update  $A \leftarrow A \cup \mathcal{D}_u + u$ .
7   otherwise Update  $T \leftarrow T - u$ .

```

---

Let  $L_\ell$  be the sum of  $f(u \mid A \cup \mathcal{D}_u)$  for all the iterations done by Algorithm 3, regardless of the random choice made by the algorithm. We show how to lower bound the expectation of  $L_\ell$  with respect to  $f(OPT)$ , and then use a bound on the variance of  $W_\ell$  to get a concentration result for  $W_\ell$ .

**Lemma 6.2.**  $(d+1) \cdot L_\ell \geq f(A_\ell)$ .

*Brief sketch of proof.* The proof is by induction on the number of iterations. Assume the lemma is true for  $i-1$  iterations, and let us prove it for iteration  $i$ . Trivially, if Algorithm 3 randomly chooses not to add elements to  $A$ , then the lemma is true for iteration  $i$  as well. Now, assume the random choice made is to add the elements to  $A$ . Note that only up to  $d+1$  elements are added to  $A$ , and therefore, it is sufficient to show that the marginal value of each is upper bounded by the increase in the value of  $L$ . Let  $u$  be the element chosen by the algorithm. Any dependency of  $u$  that appears in  $T$  could also be chosen instead of  $u$ , so its marginal is upper bounded by the increase in the value of  $L$  (since the algorithm uses a greedy choice). On the other hand, any dependency  $u' \notin T$  must have been removed when chosen by the algorithm in a previous iteration. Therefore, its marginal value, computed with respect to its optimal dependencies in this previous iteration, is already counted by  $L_\ell$ . Note that the last marginal value must be at least as large as the marginal value of  $u'$  with respect to the optimal dependencies in the current iteration (by definition of supermodular dependencies).  $\square$

**Lemma 6.3.**  $f(A_\ell) + (d+1) \cdot L_\ell \geq f(OPT)$ .

*Brief sketch of proof.* We consider a hybrid solution starting as  $OPT$  and ending as  $A_\ell$ . We use the matroid augmentation property to observe that, when a new element of  $A$  is added to this hybrid solution, no more than  $d+1$  non  $A$  elements have to be removed to restore independence. Then, we bound the “damage” resulting from the removal of these elements using the greedy choice of the algorithm and the definition of supermodular dependencies. Finally, we observe that the value of  $A_\ell$  cannot be increased by adding elements that are still in  $T$ , since, otherwise, the algorithm would have done that. This means that  $A_\ell$  itself is as good as the final hybrid (which contains it).  $\square$

An immediate corollary of the last two lemmata is a lower bound of  $f(OPT)/(2(d+1))$  on  $L_\ell$ . This bound, together with a concentration result we prove in Appendix E, shows that  $W_\ell \geq f(OPT)/O(d^2)$  with constant probability. The last inequality implies with constant probability, when  $m^* \leq f(OPT)/(256(d+1)^2)$ , that  $\text{opt}_{80(d+2)^2} = W/10$  is indeed a  $(80(d+2)^2)$ -estimation for the optimum of the part of the input that was not read by the estimation algorithm (*i.e.*, the input for the aided algorithm). The competitive ratio of the second part of Algorithm 2 then follows from the competitive ratio of the aided algorithm.

**Acknowledgment.** We are grateful to Uri Feige for valuable discussions.

## References

- [1] Ittai Abraham, Moshe Babaioff, Shaddin Dughmi, and Tim Roughgarden. Combinatorial auctions with restricted complements. In *EC*, pages 3–16, New York, NY, USA, 2012. ACM.
- [2] Pablo D. Azar, Robert Kleinberg, and S. Matthew Weinberg. Prophet inequalities with limited information. In *SODA*, pages 1358–1377, 2014.
- [3] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In Moses Charikar, Klaus Jansen, Omer Reingold, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Lecture Notes in Computer Science, pages 16–28. Springer Berlin / Heidelberg, 2007.
- [4] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exchanges*, 7(2):7:1–7:11, Jun 2008.
- [5] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, pages 434–443, 2007.
- [6] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Transactions on Algorithms*, 9(4):32, 2013.
- [7] Liad Blumrosen and Noam Nisan. On the computational power of demand queries. *SIAM Journal on Computing*, 39:1372–1391, 2009.
- [8] Sourav Chakraborty and Oded Lachish. Improved competitive ratio for the matroid secretary problem. In *SODA*, pages 1702–1712, 2012.
- [9] Nedyalko B. Dimitrov and C. Greg Plaxton. Competitive weighted matching in transversal matroids. *Algorithmica*, 62(1–2):333–348, 2012.
- [10] Michael Dinitz and Guy Kortsarz. Matroid secretary for regular and decomposable matroids. *SIAM J. Comput.*, 43(5):1807–1830, 2014.
- [11] E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Sov. Math. Dokl.*, 4, 1963.
- [12] Yuval Emek, Magnús M. Halldórsson, Yishay Mansour, Boaz Patt-Shamir, Jaikumar Radhakrishnan, and Dror Rawitz. Online set packing. *SIAM Journal on Computing*, 41(4):728–746, 2012.
- [13] Uriel Feige, Michal Feldman, Nicole Immorlica, Rani Izsak, Brendan Lucier, and Vasilis Syrgkanis. A unifying hierarchy of valuations with complements and substitutes. In *AAAI*, pages 872–878, 2015.
- [14] Uriel Feige and Rani Izsak. Welfare maximization and the supermodular degree. In *ITCS*, pages 247–256, 2013.
- [15] Moran Feldman and Rani Izsak. Constrained monotone function maximization and the supermodular degree. In *APPROX-RANDOM*, pages 160–175, 2014.

- [16] Moran Feldman, Joseph (Seffi) Naor, and Roy Schwartz. Improved competitive ratios for submodular secretary problems. In *APPROX*, pages 218–229, 2011.
- [17] Moran Feldman, Ola Svensson, and Rico Zenklusen. A simple order-oblivious  $O(\log \log(\text{rank}))$ -competitive algorithm for the matroid secretary problem. In *SODA*, pages 1189–1201, 2015.
- [18] Moran Feldman and Rico Zenklusen. The submodular secretary problem goes linear, 2015. To appear in FOCS 2015.
- [19] Shayan Oveis Gharan and Jan Vondrák. On variants of the matroid secretary problem. *Algorithmica*, 67(4):472–497, 2013.
- [20] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: offline and secretary algorithms. In *WINE*, pages 246–257. Springer-Verlag, 2010.
- [21] Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating  $k$ -set packing. *Computational Complexity*, 15(1):20–39, May 2006.
- [22] Sungjin Im and Yajun Wang. Secretary problems: Laminar matroid and interval scheduling. In *SODA*, pages 1265–1274, 2011.
- [23] Patrick Jaillet, José A. Soto, and Rico Zenklusen. Advances on matroid secretary problems: Free order model and laminar case. In *IPCO*, pages 254–265, 2013.
- [24] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA*, pages 630–631, 2005.
- [25] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *ICALP*, pages 508–520, 2009.
- [26] Oded Lachish.  $O(\log \log \text{rank})$  competitive-ratio for the matroid secretary problem. In *FOCS*, pages 326–335, 2014.
- [27] Tengyu Ma, Bo Tang, and Yajun Wang. The simulated greedy algorithm for several submodular matroid secretary problems. In *STACS*, pages 478–489, 2013.
- [28] José A. Soto. Matroid secretary problem in the random assignment model. *SIAM Journal on Computing*, 42(1):178–211, 2013.
- [29] Anita Williams Woolley, Christopher F. Chabris, Alex Pentland, Nada Hashmi, and Thomas W. Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330:265–294, 2010.

## A Assuming our set functions are normalized is without loss of generality

**Reduction 1.** *If  $ALG$  is an  $\alpha$ -competitive algorithm for the monotone matroid secretary problem under the assumption that  $f$  is normalized, then  $ALG$  is an  $\alpha$ -competitive algorithm also without this assumption.*

*Proof.* Let  $g: 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$  be the function  $g(S) = f(S) - f(\emptyset)$ . Notice that  $g$  is a non-negative monotone function and  $\mathcal{D}_f^+ = \mathcal{D}_g^+$ . Moreover, all the oracles that an algorithm for the monotone matroid secretary problem has access to return the same answers for both  $f$  and  $g$ , and thus, the algorithm produces a random set  $S$  with the same distribution when given either  $f$  or  $g$  as input. Since  $g$  is normalized, by the definition of  $ALG$ :

$$\mathbb{E}[g(S)] \geq \frac{g(OPT)}{\alpha} ,$$

where  $OPT$  is a set maximizing  $g$  (and  $f$ ). Thus:

$$\mathbb{E}[f(S)] = \mathbb{E}[g(S)] + f(\emptyset) \geq \frac{g(OPT)}{\alpha} + f(\emptyset) \geq \frac{f(OPT)}{\alpha} . \quad \square$$

## B Small rank matroids (Proof of Theorem 3.1)

In this proof we need some additional notation. The max-marginal of an element  $u$  at time  $i$  is the largest marginal value that  $u$  can contribute to a subset of the elements that arrive after time  $i$  (while keeping the subset independent). More formally, let  $\mathcal{N}_i$  be the (random) set of the first  $i$  elements that arrived, then the max-marginal of an element  $u$  at time  $i$  is:

$$\mathbf{m}_{\max}(u, i) = \max_{\substack{S \subseteq \mathcal{N} \setminus \mathcal{N}_i \\ S + u \in \mathcal{I}}} f(u \mid S) .$$

We also use  $S_{\max}(u, i)$  to denote an arbitrary set for which the maximum is obtained. Note that one can calculate both  $\mathbf{m}_{\max}(u, i)$  and  $S_{\max}(u, i)$  in  $O(2^d)$  time. We begin the proof with the following simple claim.

**Claim B.1.** *We may assume that  $n$  is dividable by any quantity  $h$  whose value is polynomial in  $k$ .*

*Proof.* Let  $n'$  be the least multiple of  $h$  which is at least as large as  $n$ . Note that  $n'$  is polynomial in  $n$  (since  $k \leq n$ ). Let  $\mathcal{N}'$  be a ground set containing the elements of  $\mathcal{N}$  and a set  $D$  of  $n' - n$  dummy elements. We extend  $f$  and  $M$  to  $\mathcal{N}'$  as follows:

- The function  $f': 2^{\mathcal{N}'} \rightarrow \mathbb{R}^+$  is defined as:  $f'(S) = f(S \setminus D)$  for every set  $S \subseteq \mathcal{N}'$ . Note that  $f'$  is non-negative, monotone and has a supermodular degree of  $d$ . Additionally,  $\mathcal{D}_{f'}^+(u) = \emptyset$  for the dummy elements of  $D$ , and  $\mathcal{D}_{f'}^+(u) = \mathcal{D}_f^+(u)$  for every other element.
- The matroid  $M' = (\mathcal{N}', \mathcal{I}')$  is defined by the following rule. A set  $S \subseteq \mathcal{N}'$  is in  $\mathcal{I}'$  if and only if  $S \setminus D \in \mathcal{I}$  and  $|S| \leq k$ . Note that this rule defines a matroid of rank  $k$  which is uniform whenever  $M$  is.

One can observe that the problems  $(f, M)$  and  $(f', M')$  are equivalent in the sense that: any solution for  $(f, M)$  is also a solution for  $(f', M')$  of the same value, and removing the dummy elements of any solution for  $(f', M')$  results in a solution for  $(f, M)$  of the same value. Moreover, given access to the oracles corresponding to  $(f, M)$ , one can efficiently implement the oracles for  $(f', M')$ . Thus, given algorithm  $ALG$  that is  $r$ -competitive for ground sets obeying the requirements of the reduction, one can construct an  $r$ -competitive algorithm for general ground sets as follows:

1. Apply  $ALG$  to the instance  $(f', M')$ .
2. Accept every element of  $\mathcal{N} = \mathcal{N}' \setminus D$  that  $ALG$  accepts. □

In the rest of this section we make two assumptions. First, we assume that  $n$  is dividable by  $10k$ , which is justified by Claim B.1. Second, we assume  $k \geq 2$  (if  $k = 1$ , then the classical secretary algorithm can be used to get an  $O(1)$ -competitive algorithm). Our objective is to show that Algorithm 4 obeys the requirements of Theorem 3.1 given these assumptions.

---

**Algorithm 4:** Small Rank Matroid

---

- 1 Select an arbitrary order  $\prec$  over the elements of the ground set  $\mathcal{N}$ .
  - 2 Let  $p$  be a uniformly random integer from the set  $\{0, 1, \dots, \lceil \log_2 k \rceil\}$ .
  - 3 Reject the first  $t = 2^p \cdot \frac{n}{2k}$  elements.
  - 4 **for**  $i = t + 1$  **to**  $n$  **do**
  - 5     Let  $u_i$  be the element arriving at time  $i$ .
  - 6     **if** for every element  $u \in \mathcal{N}_{i-1}$  either  $\mathbf{m}_{\max}(u_i, i) > \mathbf{m}_{\max}(u, i)$  or
  - 7          $(\mathbf{m}_{\max}(u_i, i) = \mathbf{m}_{\max}(u, i) \text{ and } u_i \succ u)$  **then**
  - 8         └ Terminate the “for” loop and accept the elements of  $\mathcal{S}_{\max}(u_i, i) + u_i$  when they arrive.
- 

For the purpose of analyzing Algorithm 4, it is helpful to think about the input as created backwards by the following process. The set  $\mathcal{N}_n$  is simply the entire ground set  $\mathcal{N}$ . Then, the last element of the input  $u_n$  is selected uniformly at random from  $\mathcal{N}_n$ , and the set  $\mathcal{N}_{n-1}$  becomes  $\mathcal{N}_n - u_n$ . On the next step, the  $(n-1)$ -th element  $u_{n-1}$  of the input is selected uniformly at random from  $\mathcal{N}_{n-1}$  and we set  $\mathcal{N}_{n-2} = \mathcal{N}_{n-1} - u_{n-1}$ . The process than continuous in the same way, *i.e.*, when it is time to determine the  $i$ -th element of the input, this element is selected uniformly at random from  $\mathcal{N}_i$ , and we set  $\mathcal{N}_{i-1} = \mathcal{N}_i - u_i$ .

We say that an element  $u$  is the **top** element of a set  $\mathcal{N}_i$  if for every other element  $u' \in \mathcal{N}_i \setminus u$  either  $\mathbf{m}_{\max}(u', i) < \mathbf{m}_{\max}(u, i)$  or  $\mathbf{m}_{\max}(u', i) = \mathbf{m}_{\max}(u, i)$  and  $u' \prec u$ . Note that Line 7 of Algorithm 4 in fact checks whether  $u_i$  is the top element of  $\mathcal{N}_i$ . Additionally, we say that an input is *well-behaved* with respect to the value  $t$  and order  $\prec$  chosen by Algorithm 4 if it has the following properties:

- (A1) There exists a time  $i > n/k$  such that for some element  $u \in \mathcal{N}_i$ ,  $\mathbf{m}_{\max}(u, i) \geq f(\text{OPT})/k$ , where  $\text{OPT}$  is an independent set maximizing  $f$ . We denote the first such time by  $\ell_1$ .
- (A2) There exists exactly a single time  $t < i \leq \ell_1$  such that  $u_i$  is the top element of  $\mathcal{N}_i$ .<sup>5</sup> We denote this time by  $\ell_2$ .

The analysis of Algorithm 4 consists of two parts. First we show that it produces a good output for well-behaved inputs, and then we show that the input is well-behaved with a significant probability.

**Lemma B.2.** *Algorithm 4 outputs a solution of value at least  $f(\text{OPT})/k$  when its input is well-behaved with respect to  $t$  and  $\prec$ .*

*Proof.* The definition of the algorithm and Property A2 guarantees that the algorithm outputs  $\mathcal{S}_{\max}(u_{\ell_2}, \ell_2) + u_{\ell_2}$ . The value of this solution is:

$$f(\mathcal{S}_{\max}(u_{\ell_2}, \ell_2) + u_{\ell_2}) \geq f(u_{\ell_2} \mid \mathcal{S}_{\max}(u_{\ell_2}, \ell_2)) = \mathbf{m}_{\max}(u_{\ell_2}, \ell_2) .$$

Hence, we are only left to lower bound  $\mathbf{m}_{\max}(u_{\ell_2}, \ell_2)$ . Observe that by Property A1, there exists an element  $u'_{\ell_1} \in \mathcal{N}_{\ell_1}$  such that  $\mathbf{m}_{\max}(u'_{\ell_1}, \ell_1) \geq f(\text{OPT})/k$ . Let us prove by a backward induction that this is true for every  $\ell_2 \leq i \leq \ell_1$ , *i.e.*, that for every such time there exists an element  $u'_i \in \mathcal{N}_i$  such that  $\mathbf{m}_{\max}(u'_i, i) \geq f(\text{OPT})/k$ .

Assume the claim holds for a given  $\ell_2 < i \leq \ell_1$ , and let us prove it for  $i - 1$ . Observe that we can assume without loss of generality that  $u'_i$  is the top element of  $\mathcal{N}_i$ . Thus, by Property A2

---

<sup>5</sup>Note that in some cases we might have  $\ell_1 \leq t$ . In these cases the input is not well-behaved with respect to  $t$  and  $\prec$ .

$u'_i \neq u_i$ , which implies:  $u'_i \in \mathcal{N}_{i-1}$ . By definition  $\mathbf{m}_{\max}(u, i)$  is a non-increasing function of  $i$ , hence,  $\mathbf{m}_{\max}(u'_i, i-1) \geq \mathbf{m}_{\max}(u'_i, i) \geq f(OPT)/k$ , which complete the induction step.

The claim we proved by induction implies:  $\mathbf{m}_{\max}(u'_{\ell_2}, \ell_2) \geq f(OPT)/k$ . The lemma now follows by observing that Property A2 guarantees that  $u_{\ell_2}$  is the top element of  $\mathcal{N}_{\ell_2}$ , and thus,  $\mathbf{m}_{\max}(u_{\ell_2}, \ell_2) \geq \mathbf{m}_{\max}(u'_{\ell_2}, \ell_2)$ .  $\square$

**Lemma B.3.** *Property A1 holds with a probability of at least 0.2.*

*Proof.* Given an element  $u \in \mathcal{N}$ , let  $i_u$  denote the time when it arrives. Then,

$$\sum_{u \in OPT} \mathbf{m}_{\max}(u, i_u) \geq \sum_{u \in OPT} f(u \mid OPT \setminus \mathcal{N}_u) = f(OPT) ,$$

where the inequality follows from the definition of  $\mathbf{m}_{\max}$ . Since  $|OPT| \leq k$ , we get by averaging that for some element  $u \in OPT$  there must be  $\mathbf{m}_{\max}(u, i_u) \geq f(OPT)/k$ . Hence, Property A1 is guaranteed to hold when all the elements of  $OPT$  appear after time  $\hat{t} = n/k$ . The last event occurs with a probability of at least:

$$\begin{aligned} \frac{(\hat{t}! \cdot \binom{n-k}{\hat{t}}) \cdot (n-\hat{t})!}{n!} &= \frac{(n-k)! \cdot (n-\hat{t})!}{n! \cdot (n-k-\hat{t})!} = \prod_{i=0}^{\hat{t}-1} \frac{n-k-i}{n-i} \\ &\geq \left( \frac{n-k-\hat{t}}{n-\hat{t}} \right)^{\hat{t}} = \left( 1 - \frac{k}{n-\hat{t}} \right)^{\hat{t}} \geq \left( 1 - \frac{k}{0.9n} \right)^{n/k} \\ &\geq e^{-1/0.9} \cdot \left( 1 - \frac{k}{0.9^2 \cdot n} \right) \geq e^{-10/9} \cdot \left( 1 - \frac{1}{8.1} \right) \geq 0.2 . \end{aligned} \quad \square$$

**Lemma B.4.** *Given that Property A1 holds, Property A2 holds with a probability of at least  $(\log_2 k + 2)^{-1}/4$ .*

*Proof.* First, let us consider the event  $E_1$  that there exists a time  $\ell_1/2 < \ell'_2 \leq \ell_1$  such that  $u_{\ell'_2}$  is the top element of  $\mathcal{N}_{\ell'_2}$  and for every time  $\ell'_2 < i \leq \ell_1$ ,  $u_i$  is not the top element of  $\mathcal{N}_i$ . For this event not to occur, a non-top element  $u_i$  must be selected from  $\mathcal{N}_i$  for every time  $\ell_1/2 < i \leq \ell_1$ , which happens with probability:

$$\prod_{i=\lfloor \ell_1/2+1 \rfloor}^{\ell_1} \frac{i-1}{i} = \frac{\lfloor \ell_1/2 \rfloor}{\ell_1} \leq \frac{1}{2} .$$

Hence,  $E_1$  occurs with the complement probability, which is at least  $1/2$ . Next, given that  $E_1$  occurred, we are interested in the event that  $\ell'_2/2 \leq t < \ell'_2$ , which we denote by  $E_2$ . It is important to notice that  $E_1$  is independent of the choice of  $t$  by the algorithm, and thus, the distribution of  $t$  is unaffected by conditioning on  $E_1$ . Additionally, notice that:

$$2^0 \cdot \frac{n}{2k} = \frac{n}{2k} \leq \frac{\ell_1}{2} < \ell'_2 \quad \text{and} \quad \frac{\ell'_2}{2} \leq \frac{n}{2} \leq 2^{\lceil \log_2 k \rceil} \cdot \frac{n}{2k} .$$

Hence, one of the possible values of  $t$  obeys the requirement  $\ell'_2/2 \leq t < \ell'_2$ . Since  $t$  takes at most  $\log_2 k + 2$  different values, and it takes them with equal probabilities, we get that  $E_2$  occurs with a probability of at least  $(\log_2 k + 2)^{-1}$  given  $E_1$ .

Given that  $E_1$  and  $E_2$  both occur, for Property A2 to hold we need the additional event that in the range  $(t, \ell'_2)$  no element  $u_i$  is the top element of  $\mathcal{N}_i$ . Note that the order of the elements of  $\mathcal{N}_{\ell'_2} - u_{\ell'_2}$  is independent of  $E_1$  and  $E_2$ . Hence, the probability of this event is at least:

$$\prod_{i=t+1}^{\ell'_2-1} \frac{i-1}{i} = \frac{t}{\ell'_2-1} \geq \frac{\ell'_2/2}{\ell'_2-1} > 1/2 . \quad \square$$



We are now ready to prove Theorem 3.1.

*Proof of Theorem 3.1.* Lemmata B.3 and B.4 imply that the input is well-behaved with respect to  $t$  and  $\prec$  with probability at least  $(\log_2 k + 2)^{-1}/20$ . By Lemma B.2, when the input is well-behaved with respect to  $t$  and  $\prec$ , Algorithm 4 outputs a solution of value at least  $f(OPT)/k$ . Hence, the competitive ratio of Algorithm 4 is at least:

$$20k(\log_2 k + 2) = O(k \log k) . \quad \square$$

## C Missing proofs

This section contains proofs that have been omitted from the main body of the paper.

*Proof of Lemma 5.2.* Clearly  $w(OPT) = f(OPT)$ . Moreover, by definition,  $2^{p_1} \cdot \text{opt}_\alpha \leq f(OPT)/k$ . Hence:

$$f(OPT) - w\left(\bigcup_{p=p_1}^{p_2} OPT_p\right) = \sum_{\substack{w \in OPT \\ w(u) < 2^{p_1} \cdot \text{opt}_\alpha / 2}} w(u) \leq k \cdot (2^{p_1} \cdot \text{opt}_\alpha / 2) \leq \frac{f(OPT)}{2} . \quad \square$$

*Proof of Lemma 5.3.* For an element  $u \in \mathcal{N}$ , let  $S_u$  be the set  $S$  immediately before  $u$  is processed by Algorithm 1. Note that each time that Algorithm 1 adds elements to  $S$ , it adds up to  $d + 1$  elements and  $f(S)$  increases by at least  $\tau$  since, by monotonicity:

$$f(D^*(u) + u \mid S_u) \geq f(u \mid D^*(u) \cup S_u) .$$

Hence, we can lower bound  $f(S)$  by:

$$f(S) \geq \left\lceil \frac{|S|}{d+1} \right\rceil \cdot \tau \geq \frac{|OPT_{p+3}|}{2(d+1)^2} \cdot \left\lceil \frac{1}{16} \cdot \max_{u \in OPT_{p+3}} w(u) \right\rceil \geq \frac{w(OPT_{p+3})}{32(d+1)^2} . \quad \square$$

*Proof of Lemma 5.4.* Observe that  $OPT_{p+3}$  is a subset of  $OPT$ , and thus, independent. Hence, by the matroid properties, there exists a set  $O' \subseteq OPT_{p+3} \setminus S$  of size at least  $|OPT_{p+3}| - |S|$  such that  $O' \cup S \in \mathcal{I}$ . Every element  $u \in OPT_{p+3} \setminus O'$  can belong to the dependence set of at most  $d$  other elements of  $OPT_{p+3}$ . Thus, the number of elements  $u \in OPT_{p+3}$  having  $\mathcal{D}^+(u) \cap OPT + u \not\subseteq O'$  is upper bounded by:

$$(d+1) \cdot |S| < \frac{|OPT_{p+3}|}{2} .$$

In other words, there exists a set  $O'' \subseteq OPT_{p+3}$  of size at least  $|OPT_{p+3}|/2$  such that  $\mathcal{D}^+(u) \cap OPT + u \subseteq O'$  for every  $u \in O''$ . Observe that by monotonicity:

$$\begin{aligned} f(O') &\geq \sum_{u \in O''} f(u \mid O' \setminus \mathcal{N}_u) \geq \sum_{u \in O''} f(u \mid OPT \setminus \mathcal{N}_u) \\ &= w(O'') \geq \frac{|OPT_{p+3}|}{2} \cdot \min_{u \in OPT_{p+3}} w(u) \geq \frac{w(OPT_{p+3})}{4} , \end{aligned}$$

where the second inequality holds since  $O'$  already contains all the elements of  $\mathcal{D}^+(u) \cap OPT$ .

Every element  $u \in O'$  must have been rejected upon arrival by Algorithm 1 due to the threshold. Moreover, for every such element  $u$  we have  $([\mathcal{D}^+(u) \cap (O' \cup S)] \setminus \mathcal{N}_u + u) \cup S_u \subseteq O' \cup S \in \mathcal{I}$  (where  $S_u$  is, again, the set  $S$  immediately before  $u$  is processed by Algorithm 1). Hence:

$$f(u \mid (O' \setminus \mathcal{N}_u) \cup S) \leq f(u \mid [\mathcal{D}^+(u) \cap (O' \cup S)] \setminus \mathcal{N}_u \cup S_u) < \tau ,$$

where the first inequality holds by the definition of  $\mathcal{D}^+(u)$ . Adding the last inequality over all elements  $u \in O'$  gives:

$$\begin{aligned} \frac{w(OPT_{p+3})}{4} &\leq f(O') \leq f(O' \cup S) = f(S) + \sum_{u \in O'} f(u \mid (O' \setminus \mathcal{N}_u) \cup S) < f(S) + |OPT_{p+3}| \cdot \tau \\ &\leq f(S) + |OPT_{p+3}| \cdot \frac{\min_{u \in OPT_{p+3}} w(u)}{8} \leq f(S) + \frac{w(OPT_{p+3})}{8}. \end{aligned}$$

The lemma now follows by rearranging the last inequality.  $\square$

**Lemma C.1.** *If  $m^* \geq f(OPT)/[256(d+1)^2]$ , then Algorithm 2 is  $O(d^3 \log d)$ -competitive.*

*Proof.* Note that  $S^* + u^*$  is an independent set in the matroid even after it is truncated to rank  $\min\{k, d+1\}$ . Hence, when Algorithm 2 applies the algorithm guaranteed by Theorem 3.1 (which happens with probability  $1/2$ ), the produced set has an expected value of at least:

$$\frac{f(S^* + u^*)}{O(d \log d)} \geq \frac{f(u^* \mid S^*)}{O(d \log d)} = \frac{m^*}{O(d \log d)} \geq \frac{f(OPT)/[256(d+1)^2]}{O(d \log d)}. \quad \square$$

## D Estimation aided algorithm for a uniform matroid constraint

In this section we prove the following theorem.

**Theorem D.1.** *For every  $\alpha \geq 1$ , there exists an  $\alpha$ -aided  $O(d \log \alpha)$ -competitive algorithm for the monotone matroid secretary problem when the matroid  $M$  is uniform.*

Before proving the existence of an  $\alpha$ -aided algorithms for any  $\alpha \geq 1$ , let us begin with a 2-aided algorithm.

**Proposition D.2.** *There exists a 2-aided  $O(d)$ -competitive algorithm for the monotone matroid secretary problem when the matroid  $M$  is uniform.*

The algorithm we use to prove Proposition D.2 is Algorithm 5.

---

### Algorithm 5: 2-Aided Cardinality

---

```

1 Let  $\tau \leftarrow \frac{\text{opt}_2}{2k}$ .
2 Let  $S \leftarrow \emptyset$ .
3 for every arriving element  $u$  do
4   if there exists a set  $D^*(u) \subseteq \mathcal{D}^+(u) \setminus \mathcal{N}_u$  such that  $f(u \mid D^*(u) \cup S) \geq \tau$  and
5      $|S| + |D^*(u)| + 1 \leq k$  then
6      $\quad$   $\lfloor$  Add  $D^*(u) + u$  to  $S$ .
7 return  $S$ .
```

---

Let  $S_u$  be the set  $S$  before the element  $u$  is processed. When  $S$  appears below without a subscript it denotes the output of the algorithm.

**Lemma D.3.** *If  $|S| \leq \max\{0, k - d - 1\}$ , then  $f(S) \geq f(OPT)/2$ .*

*Proof.* Assume, towards a contradiction, that  $|S| \leq \max\{0, k - d - 1\}$  and still  $f(S) < f(OPT)/2$ . Since  $|S| \leq \max\{0, k - d - 1\}$ , for every element  $u \in OPT \setminus S$  Algorithm 5 could add the set  $[\mathcal{D}^+(u) \cap (OPT \cup S)] \setminus \mathcal{N}_u + u$  to  $S$ . From the fact that the algorithm did not add this set (or any other set containing  $u$ ) to  $S$ , we learn that:

$$f(u \mid (OPT \setminus \mathcal{N}_u) \cup S) \leq f(u \mid [\mathcal{D}^+(u) \cap (OPT \cup S)] \setminus \mathcal{N}_u \cup S_u) < \tau ,$$

where the first inequality holds by the definition of  $\mathcal{D}^+(u)$ . Adding the last inequality over all elements  $u \in OPT \setminus S$  gives:

$$f(OPT) \leq f(OPT \cup S) = f(S) + \sum_{u \in OPT} f(u \mid (OPT \setminus \mathcal{N}_u) \cup S) < f(S) + k\tau .$$

Plugging the assumption that  $f(S) < f(OPT)/2$  and the definition of  $\tau$  into the last inequality gives an immediate contradiction.  $\square$

**Lemma D.4.** *If  $|S| \geq \max\{1, k - d\}$ , then  $f(S) \geq f(OPT)/[8(d + 1)]$ .*

*Proof.* Note that each time that Algorithm 5 adds elements to  $S$ , it adds up to  $d + 1$  elements and  $f(S)$  increases by at least  $\tau$  since, by monotonicity:

$$f(D^*(u) + u \mid S_u) \geq f(u \mid D^*(u) \cup S_u) .$$

Hence, we can lower bound  $f(S)$  by:

$$f(S) \geq \left\lceil \frac{|S|}{d + 1} \right\rceil \cdot \tau \geq \left\lceil \frac{\max\{1, k - d\}}{d + 1} \right\rceil \cdot \frac{\text{opt}_2}{2k} \geq \frac{k}{2(d + 1)} \cdot \frac{f(OPT)}{4k} = \frac{f(OPT)}{8(d + 1)} . \quad \square$$

Proposition D.2 follows immediately from the last two lemmata. Theorem D.1 generalizes Proposition D.2 to general  $\alpha$ -aided algorithms. The algorithm we use to prove Theorem D.1 is Algorithm 6.

---

**Algorithm 6:**  $\alpha$ -Aided Cardinality

---

- 1 Let  $p$  be a uniformly random integer from the set  $\{0, 1, \dots, \lceil \log_2 \alpha \rceil\}$ .
  - 2 Apply Algorithm 5 with  $\text{opt}_2 = 2^p \cdot \text{opt}_\alpha$ .
- 

*Proof of Theorem D.1.* The largest value Algorithm 6 can assign to  $\text{opt}_2$  is:

$$2^{\lceil \log \alpha \rceil} \cdot \text{opt}_\alpha \geq \alpha \cdot (f(OPT)/\alpha) = f(OPT) .$$

On the other hand, the smallest value Algorithm 6 can assign to  $\text{opt}_2$  is:  $2^0 \cdot \text{opt}_\alpha \leq f(OPT)$ . Hence, for some  $p$  Algorithm 6 is guaranteed to produce a value  $\text{opt}_2$  obeying  $f(OPT)/2 \leq \text{opt}_2 \leq f(OPT)$ , in which case Algorithm 5 is  $O(d)$ -competitive by Proposition D.2. Since every value of  $p$  occurs with a probability of at least  $1/(\log_2 \alpha + 2)$ , we get that the competitive ratio of Algorithm 6 is at most  $O(d \log \alpha)$ .  $\square$

## E Full proof for $m^* \leq f(OPT)/(256(d + 1)^2)$

In this section we analyze Algorithm 2 in the case of a small  $m^*$ . We begin with a concentration result proved in Section E.1. The analysis of Algorithm 2 appears in Section E.2.

## E.1 Concentration result

In this section we study a stochastic process consisting of rounds. In each round  $i \geq 1$ , a positive value  $X_i \in (0, B]$  (for some parameter  $B > 0$ ) arrives, and is flagged “accepted” with a probability  $p \in [0, 1]$ , independently, and “rejected” otherwise. The value  $X_i$  itself might depend on the way previous values have been flagged, but not on the way  $X_i$  itself is flagged. More formally, let  $A$  be the set of indexes corresponding to accepted values, then  $X_i$  is a function of the set  $A \cap \{1, 2, \dots, i-1\}$ . The process terminates after  $T \geq 1$  rounds, where  $T$  itself might depend on the way values have been flagged. However, it is guaranteed that  $T$  is upper bounded by a finite integer  $\bar{T}$  and:

$$\sum_{i=1}^T X_i \geq L$$

for some parameter  $L \geq 0$ .

Let  $\Sigma_A = \sum_{i \in A} X_i$  be the random sum of the accepted values. Our objective is to show a concentration bound for  $\Sigma_A$ . Let us first prove such a bound for the case when we make an additional assumption.

**Assumption E.1.** *Each value  $X_i$  is equal to  $B/2^j$  for some value  $j \geq 0$ .*

Using the above assumption, we can now define some additional notation. Let  $\delta$  be the smallest number such that some value  $X_i$  has a positive probability to take the value  $\delta$ . Observe that  $\delta$  is well defined since the above process has only finitely many possible outcomes. By Assumption E.1, every value  $X_i$  is a multiple of  $\delta$ .

It is helpful to think of the values  $X_i$  as intervals placed one after the other on the axis of real numbers, starting from 0. In other words, for every value  $X_i$  we have an interval starting at  $\sum_{j=1}^{i-1} X_j$  and ending at  $\sum_{j=1}^i X_j$ . Taking this point of view, every interval  $X_i$  can be partitioned into  $X_i/\delta$  ranges of size  $\delta$ . Let us associate a random variable with each one of these ranges. More formally, for every  $i \geq 1$ , let  $Y_i$  be a random variable taking the value 1 when the range  $(\delta(i-1), \delta i)$  is contained within an accepted interval  $X_j$ , and the value 0 in all other cases.

**Lemma E.2.** *Under Assumption E.1,  $\Sigma_A = \delta \cdot \sum_{i=1}^{\bar{T} \cdot B/\delta} Y_i$ .*

*Proof.* Fix a realization of the above process. Recall that  $Y_i$  is zero whenever the range  $(\delta(i-1), \delta i)$  is not contained within an accepted interval. On the other hand, consider an arbitrary accepted interval  $X_i$ . The interval  $X_i$  contains  $X_i/\delta$  ranges of size  $\delta$ . Moreover, all these ranges end at the point  $\sum_{j=1}^T X_j \leq \bar{T} \cdot B$  or earlier, and thus, their variables appear in the sum on the right hand side of the equality we want to prove. Hence, in conclusion, the contribution of  $X_i$  to that sum is exactly  $X_i/\delta$ . The observation now follows since the intervals  $\{X_i\}_{i=1}^T$  are disjoint, and thus, so are their contributions to the sum.  $\square$

Let  $I$  be the minimal integer such that  $\delta I \geq L$ .

**Observation E.3.** *Under Assumption E.1,  $\Sigma_A \geq \delta \cdot \sum_{i=1}^I Y_i$ .*

*Proof.* Notice that  $\bar{T} \cdot B$  is an upper bound on the sum  $\sum_{j=1}^T X_j$ . On the other hand,  $L$  is a lower bound on this sum, and thus, we get:  $\bar{T} \cdot B \geq L$ . Hence,  $\delta(\bar{T} \cdot B/\delta) \geq L$ . Since the term  $\bar{T} \cdot B/\delta$  is an integer, the minimality of  $I$  implies  $I \leq \bar{T} \cdot B/\delta$ . Using Lemma E.2 and the fact that the variables  $Y_i$  are non-negative, we get:

$$\Sigma_A = \delta \cdot \sum_{i=1}^{\bar{T} \cdot B/\delta} Y_i \geq \delta \cdot \sum_{i=1}^I Y_i . \quad \square$$

Observation E.3 shows that it is enough for our purpose to prove a concentration bound for  $\sum_{i=1}^I Y_i$ . The following observation gives another useful property of  $I$ .

**Observation E.4.** *Under Assumption E.1, for every  $1 \leq i \leq I$ , the range  $(\delta(i-1), \delta i)$  is always contained within some interval  $X_j$ .*

*Proof.* Assume towards a contradiction that there is some realization of the process under which the range  $(\delta(i-1), \delta i)$  is not contained within some interval  $X_j$ . This implies:

$$L \leq \sum_{j=1}^T X_j \leq \delta(i-1) \leq \delta(I-1) ,$$

contradicting the definition of  $I$ . □

Let us now study the distribution of the variables  $\{Y_i\}_{i=1}^I$ .

**Lemma E.5.** *Under Assumption E.1,  $\Pr[Y_i = 1] = p$  for every  $1 \leq i \leq I$ . Hence, by linearity of expectation:*

$$\mathbb{E} \left[ \sum_{i=1}^I Y_i \right] = \sum_{i=1}^I \mathbb{E}[Y_i] = pI .$$

*Proof.* For every  $j \geq 1$ , let  $\mathcal{E}_j$  be the event that  $j \leq T$  and  $(\delta(i-1), \delta i)$  is included in the interval  $X_j$ . Observe that  $\mathcal{E}_j$  depends only the acceptance of intervals  $X_{j'}$  for  $j' < j$ . Moreover, given that  $X_j$  exists it is accepted with probability  $p$ , independently of the acceptance of previous intervals. Hence, we get:

$$\Pr[Y_i = 1 \mid \mathcal{E}_j] = p .$$

The observation now follows by the law of total probability since Observation E.4 guarantees that  $(\delta(i-1), \delta i)$  is included in some interval, and thus, the event  $\mathcal{E}_j$  happens for exactly a single value of  $j$ . □

For every  $1 \leq h \leq B/\delta$ , let us define  $V_h = \{1 \leq i \leq I \mid i \equiv h \pmod{B/\delta}\}$ . Observe that the sets  $\{V_h\}_{h=1}^{B/\delta}$  form a disjoint partition of the indexes from 1 to  $I$ .

**Observation E.6.** *Under Assumption E.1, for every  $1 \leq h \leq B/\delta$  and two different indexes  $i, i' \in V_h$ , the ranges  $(\delta(i-1), \delta i)$  and  $(\delta(i'-1), \delta i')$  cannot be contained in one interval  $X_j$ .*

*Proof.* Assume without loss of generality that  $i < i'$ . The definition of  $V_h$  guarantees that  $i + B/\delta \leq i'$ . Hence,

$$\delta i' - \delta(i-1) = \delta(i' - i) + \delta \geq B + \delta .$$

Hence, any interval  $X_j$  containing both ranges  $(\delta(i-1), \delta i)$  and  $(\delta(i'-1), \delta i')$  must be of length at least  $B + \delta$ , which contradicts the definition of the process. □

**Lemma E.7.** *Under Assumption E.1, for every  $1 \leq h \leq B/\delta$ , the variables of  $\{Y_i \mid i \in V_h\}$  are independent.*

*Proof.* For every  $i \in V_h$ , let  $V_h^{<i}$  denote the intersection  $V_h \cap \{1, 2, \dots, i-1\}$ . To prove the lemma it is enough to show that for every  $i \in V_h$  the variable  $Y_i$  takes the value 1 with probability  $p$  conditioned on any assignment to the variables of  $\{Y_j \mid j \in V_h^{<i}\}$ . Let  $\mathcal{A}$  denote an arbitrary such assignment having a non-zero probability. For every  $1 \leq \ell \leq T$ , let  $\mathcal{E}_\ell$  be the event that the interval  $X_\ell$  exists and contains the range  $(\delta(i-1), \delta i)$ .

Assume  $\mathcal{E}_\ell$  happens. By Observation E.6 the variables of  $\{Y_j \mid j \in V_h^{<i}\}$  correspond to ranges contained in intervals before  $X_\ell$ . Thus, the values of these variables only imply information about the acceptance of these intervals. Since the acceptance of  $X_\ell$  is independent of the acceptance of previous intervals, we get that every  $1 \leq \ell \leq \bar{T}$  obeying  $\Pr[\mathcal{E}_\ell \mid \mathcal{A}] > 0$  must also obey:

$$\Pr[Y_i = 1 \mid \mathcal{E}_\ell, \mathcal{A}] = p .$$

Clearly the events  $\{\mathcal{E}_\ell\}_{\ell=1}^{\bar{T}}$  are disjoint. By Observation E.4 we also know that one of them must happen. Hence, we get:

$$\Pr[Y_i \mid \mathcal{A}] = \sum_{\substack{1 \leq \ell \leq \bar{T} \\ \Pr[\mathcal{E}_\ell \mid \mathcal{A}] > 0}} \Pr[\mathcal{E}_\ell \mid \mathcal{A}] \cdot \Pr[Y_i = 1 \mid \mathcal{E}_\ell, \mathcal{A}] = p \cdot \sum_{\ell=1}^{\bar{T}} \Pr[\mathcal{E}_\ell \mid \mathcal{A}] = p . \quad \square$$

**Corollary E.8.** *Under Assumption E.1, for every  $1 \leq h \leq B/\delta$ ,  $\text{Var} \left[ \sum_{i \in V_h} Y_i \right] \leq p(L/B + 2)$ .*

*Proof.* By Lemma E.5, for every  $i \in V_h$ ,  $\text{Var}[Y_i] = p(1-p) \leq p$ . Thus, by Lemma E.7,

$$\text{Var} \left[ \sum_{i \in V_h} Y_i \right] = \sum_{i \in V_h} \text{Var}[Y_i] \leq \sum_{i \in V_h} p = p \cdot |V_h| .$$

The definition of  $V_h$  guarantees that its size is at most:

$$|V_h| \leq \left\lceil \frac{I}{B/\delta} \right\rceil \leq \left\lceil \frac{L/\delta + 1}{B/\delta} \right\rceil \leq \frac{L + \delta}{B} + 1 \leq \frac{L}{B} + 2 . \quad \square$$

To bound the variance of the sum  $\sum_{i=1}^I Y_i$ , we need the following simple technical lemma.

**Lemma E.9.** *For a set of random variables  $Z_1, Z_2, \dots, Z_\ell$ , each having a finite variance,*

$$\text{Var} \left[ \sum_{i=1}^{\ell} Z_i \right] \leq \left( \sum_{i=1}^{\ell} \sqrt{\text{Var}[Z_i]} \right)^2 .$$

*Proof.*

$$\text{Var} \left[ \sum_{i=1}^{\ell} Z_i \right] = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \text{Cov}[Z_i, Z_j] \leq \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \sqrt{\text{Var}[Z_i] \cdot \text{Var}[Z_j]} = \left( \sum_{i=1}^{\ell} \sqrt{\text{Var}[Z_i]} \right)^2 . \quad \square$$

**Corollary E.10.** *Under Assumption E.1,  $\text{Var} \left[ \sum_{i=1}^I Y_i \right] \leq pB\delta^{-2}(L + 2B)$ .*

*Proof.* Observe that:

$$\sum_{i=1}^I Y_i = \sum_{h=1}^{B/\delta} \sum_{i \in V_h} Y_i .$$

Hence, by Corollary E.8 and Lemma E.9:

$$\text{Var} \left[ \sum_{i=1}^I Y_i \right] \leq \left( \frac{B}{\delta} \cdot \sqrt{p \left( \frac{L}{B} + 2 \right)} \right)^2 = \frac{pB}{\delta^2} (L + 2B) . \quad \square$$

We are now ready to prove the promised concentration bound for  $\Sigma_A$ .

**Lemma E.11.** *Under Assumption E.1, for every  $t > 0$ ,  $\Pr[\Sigma_A < pL - t] \leq pB(L + 2B)/t^2$ .*

*Proof.* By Lemma E.3,

$$\begin{aligned} \Pr[\Sigma_A < pL - t] &\leq \Pr\left[\delta \cdot \sum_{i=1}^I Y_i < pL - t\right] \\ &\leq \Pr\left[\delta \cdot \sum_{i=1}^I Y_i < \delta \cdot pI - t\right] \leq \Pr\left[\left|\delta \cdot \sum_{i=1}^I Y_i - \delta \cdot pI\right| > t\right]. \end{aligned}$$

Since the expected value of  $\delta \cdot \sum_{i=1}^I Y_i$  is  $\delta \cdot pI$  by Lemma E.5, we get by Chebyshev's inequality:

$$\Pr\left[\left|\delta \cdot \sum_{i=1}^I Y_i - \delta \cdot pI\right| > t\right] \leq \frac{\text{Var}\left[\delta \cdot \sum_{i=1}^I Y_i\right]}{t^2} = \frac{\delta^2 \cdot \text{Var}\left[\sum_{i=1}^I Y_i\right]}{t^2} \leq \frac{pB(L + 2B)}{t^2},$$

where the last inequality holds by Corollary E.10.  $\square$

Finally, we would like to get a version of Lemma E.11 that holds without Assumption E.1.

**Corollary E.12.** *For every  $t > 0$ ,  $\Pr[\Sigma_A < pL/2 - t] \leq pB(L + 2B)/(4t^2)$*

*Proof.* For every value  $X_i$ , let us define a value  $X'_i$  as follows:

$$X'_i = B/2^{\lfloor \log_2(B/X_i) \rfloor}.$$

Intuitively,  $X'_i$  is the smallest value allowed by Assumption E.1 that is at least as large as  $X_i$ . We say that  $X'_i$  is accepted if and only if  $X_i$  is. One can verify that the following holds:

- The values  $X'_1, X'_2, \dots, X'_T$  define a legal process with the same parameters  $p, B$  and  $L$  as the original process, and this process obeys Assumption E.1. Let  $\Sigma'_A$  be the sum of the accepted values in this process.
- For every value  $X_i$ ,  $X'_i \leq 2X_i$ , hence,  $\Sigma'_A \leq 2 \cdot \Sigma_A$ .

Thus, by Lemma E.11:

$$\Pr[\Sigma_A < pL/2 - t] \leq \Pr[\Sigma'_A < pL - 2t] \leq \frac{pB(L + 2B)}{(2t)^2} = \frac{pB(L + 2B)}{4t^2}. \quad \square$$

## E.2 Proof for $m^* \leq f(OPT)/(256(d + 1)^2)$

In this section we prove that Algorithm 2 is  $O(\beta)$ -competitive when  $m^* \leq f(OPT)/(256(d + 1)^2)$ . Recall that Algorithm 2 consists of two parts. Let us say that Algorithm 2 “applies the second option” when it executes the second part (the one that involves the aided algorithm). Notice that it is enough to show that the algorithm is  $O(\beta)$ -competitive when it applies the second option, since this option is applied with probability  $1/2$ .

We need some additional notation. First, we denote by  $\ell + 1$  the number of iterations performed by the loop on Line 6 of the algorithm (*i.e.*, the  $\ell + 1$  iteration is the iteration at which the algorithm decides to leave the loop, and does not change  $A$ ). Additionally, for every  $1 \leq i \leq \ell$ , let  $A_i$  and  $W_i$  denote the set  $A$  and the value  $W$ , respectively, immediately after the  $i$ -th iteration of this loop. For consistency, we also denote by  $A_0$  and  $W_0$  these set and value before the first iteration. Finally, for every  $1 \leq i \leq \ell$ , let  $u_i$  denote the element  $u$  chosen at iteration  $i$  of the loop. We begin the analysis of the small  $m^*$  case by proving an upper bound on  $W_\ell$  (the final value of  $W$ ).

**Observation E.13.** When Algorithm 2 applies the second option,  $A_i \in \mathcal{I}$  for every  $0 \leq i \leq \ell$ .

*Proof.* The observation holds since Algorithm 2 chooses at every iteration an element  $u$  and a set  $\mathcal{D}_u$  whose addition to  $A$  does not violate independence.  $\square$

**Lemma E.14.** When Algorithm 2 applies the second option,  $W_\ell \leq f(OPT)$ .

*Proof.* We prove by induction on  $i$  the claim that the inequality  $W_i \leq f(A_i)$  holds for every  $0 \leq i \leq \ell$ . Notice that the observation follows from this claim since, by Observation E.13,  $A_\ell$  is independent, and thus,  $f(A_\ell) \leq f(OPT)$ .

For  $i = 0$  the claim is trivial since  $W_0 = 0 = f(\emptyset) = f(A_0)$ . For  $i > 0$ , assume the claim holds for  $i - 1$ , and let us prove it for  $i$ .

$$\begin{aligned} W_i &= W_{i-1} + f(u_i \mid A_{i-1} \cup \mathcal{D}_{u_i}) \leq f(A_{i-1}) + f(u_i \mid A_{i-1} \cup \mathcal{D}_{u_i}) \\ &\leq f(A_{i-1} \cup \mathcal{D}_{u_i}) + f(u_i \mid A_{i-1} \cup \mathcal{D}_{u_i}) = f(A_i) , \end{aligned}$$

where the first inequality holds by the induction hypothesis, and the second inequality follows from the monotonicity of  $f$ .  $\square$

Our next objective is to prove a lower bound on  $W_\ell$  that holds with a constant probability. For that purpose, let us consider an offline algorithm which calculates a value  $W$  having the same distribution as the value  $W$  calculated by Algorithm 2.

---

**Algorithm 7:** Offline  $W$  Calculation

---

```

1 Let  $A \leftarrow \emptyset$ ,  $W \leftarrow 0$  and  $T \leftarrow \mathcal{N}$ .
2 while there exist  $u \in T \setminus A$  and  $\mathcal{D}_u \subseteq \mathcal{D}^+(u)$  s.t.  $A \cup \mathcal{D}_u + u \in \mathcal{I}$  do
3   Find such a pair maximizing  $f(u \mid A \cup \mathcal{D}_u)$ .
4   with probability  $(d + 2)^{-1}$  do
5     Increase  $W \leftarrow W + f(u \mid A \cup \mathcal{D}_u)$ .
6     Update  $A \leftarrow A \cup \mathcal{D}_u + u$ .
7   otherwise Update  $T \leftarrow T - u$ .
```

---

**Observation E.15.** The distribution of the value  $W$  calculated by Algorithm 3 is identical to the distribution of  $W_\ell$  as calculated by Algorithm 2 when it applies the second option.

*Proof.* The loop starting on Line 6 of Algorithm 2 looks in each iteration for a pair of an element  $u$  and a set  $\mathcal{D}_u$  maximizing  $f(u \mid A \cup \mathcal{D}_u)$  and obeying some other conditions, including the requirement that  $u$  belongs to a set  $T$  containing every element with probability  $(d + 2)^{-1}$ , independently.

On the other hand, Algorithm 3 has a loop that looks for a pair of an element  $u$  and a set  $\mathcal{D}_u$  maximizing  $f(u \mid A \cup \mathcal{D}_u)$  and obeying the same conditions, except for requiring  $u$  to belong to  $T$ . Once such a pair is found, the algorithm makes a random decision whether to keep  $u$  in  $T$ , and then uses the pair to increase the solution  $A$  if and only if it decides to keep  $u$  in  $T$ .

Notice that the only difference between these two procedures is the point when the algorithm decides whether each element  $u$  should belong to  $T$ . Algorithm 2 makes the decisions for all elements at the beginning, while Algorithm 3 makes the decisions only when necessary. However, regardless of when the membership of elements in  $T$  is decided, the set of pairs used to increase the solution  $A$  is the same given that the same random decisions are made by both algorithms.  $\square$

To analyze the distribution of the value  $W$  calculated by Algorithm 3 we need some additional notation. Let  $\hat{\ell} + 1$  be the number of iterations performed by the loop of Algorithm 3 (*i.e.*,  $\hat{\ell}$  is the



number of times elements are added to the set  $A$ ), and for every  $1 \leq i \leq \hat{\ell}$  let  $\hat{A}_i$ ,  $\hat{T}_i$  and  $\hat{W}_i$  denote the sets  $A$  and  $T$  and the value  $W$ , respectively, immediately after the  $i$ -th iteration of this loop. For consistency, we also denote by  $\hat{A}_0$ ,  $\hat{T}_0$  and  $\hat{W}_0$  these sets and value before the first iteration. Additionally, for every  $1 \leq i \leq \hat{\ell}$ , let  $\hat{u}_i$  denote the element  $u$  chosen at iteration  $i$  of the loop. Finally, for every  $0 \leq i \leq \hat{\ell}$ , let us denote by  $OPT_i$  the maximum value independent set that can be obtained from  $\hat{A}_i$  by adding only  $T$  elements. Formally,

$$OPT_i = \arg \max_{S \in \mathcal{I} | \hat{A}_i \subseteq S \subseteq \hat{A}_i \cup \hat{T}_i} f(S) .$$

Observe that  $OPT_i$  is well defined since the set  $\hat{A}_i$  is independent for every  $0 \leq i \leq \hat{\ell}$ .

Next, observe that Algorithm 3 can be viewed as a process of the kind described in Section E.1. More precisely, each iteration  $i$  of Algorithm 3 corresponds to one iteration of the process, the value of this iteration is  $f(\hat{u}_i | \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$  and this value is accepted if and only if  $\hat{u}_i$  is kept in  $T$  (and  $f(\hat{u}_i | \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$  is added to  $W$ ). Note that, as required by the process definition, the value  $f(\hat{u}_i | \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$  depends only on the acceptances of previous values, and the acceptances of the value  $f(\hat{u}_i | \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$  is independent of anything else. Taking this point of view,  $W_{\hat{\ell}}$  is exactly the sum of the accepted values, and thus, can be analyzed using Corollary E.12. To use the last corollary, we need to determine the parameters of the process:

- By definition,  $m^*$  upper bounds  $f(\hat{u}_i | (\hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) \cap \mathcal{D}^+(u)) \geq f(\hat{u}_i | \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$ . Hence, one can choose  $B = f(OPT)/[256(d+1)^2] \geq m^*$  for the process given by Algorithm 3.
- Every value is accepted with probability  $(d+2)^{-1}$ , thus, this is the value of  $p$ .

We are left to determine a possible value for the parameter  $L$ . For that purpose we need a few claims.

**Observation E.16.**  $OPT_{\hat{\ell}} = \hat{A}_{\hat{\ell}}$ .

*Proof.* By definition  $OPT_{\hat{\ell}}$  contains the elements of  $\hat{A}_{\hat{\ell}}$  and (possibly) additional elements of  $\hat{T}_{\hat{\ell}}$  that do not violate independence when added to  $\hat{A}_{\hat{\ell}}$ . However, the fact that Algorithm 3 stopped during the  $\hat{\ell} + 1$  iteration implies that no elements of  $\hat{T}_{\hat{\ell}}$  can be added to  $\hat{A}_{\hat{\ell}}$  without violating independence. Therefore,  $OPT_{\hat{\ell}}$  cannot contain any elements beside the elements of  $\hat{A}_{\hat{\ell}}$ .  $\square$

For every  $0 \leq i \leq \hat{\ell}$ , let  $L_i$  be the sum of the first  $i$  values of the process corresponding to Algorithm 3. Formally,

$$L_i = \sum_{j=1}^i f(\hat{u}_j | \hat{A}_{j-1} \cup \mathcal{D}_{\hat{u}_j}) .$$

**Lemma E.17.** For every  $0 \leq i \leq \hat{\ell}$ ,  $f(\hat{A}_i) \leq (d+1) \cdot L_i$ .

*Proof.* We prove by induction on  $i$  that for every  $0 \leq i \leq \hat{\ell}$ :

$$f(\hat{A}_i) \leq (d+1) \cdot \sum_{\hat{u}_j \in \hat{A}_i} f(\hat{u}_j | \hat{A}_{j-1} \cup \mathcal{D}_{\hat{u}_j}) . \quad (1)$$

Observe that the lemma follows from the last claim since  $\hat{u}_j$  can belong to the set  $\hat{A}_i$  only when  $j \leq i$ . For  $i = 0$ , Equation (1) is trivial since  $f(\hat{A}_0) = f(\emptyset) = 0$ . Next, assume Equation (1) holds for  $i - 1 \geq 0$ , and let us prove it for  $i$ . If  $\hat{u}_i$  is removed from  $T$  then this is true since in this case  $\hat{A}_i = \hat{A}_{i-1}$ . Thus, we can safely assume in the rest of the proof that  $\hat{u}_i$  remains in  $T$ .

For every  $0 \leq i \leq \hat{\ell}$ , let  $N_i = \{\hat{u}_j \notin \hat{A}_i \mid 1 \leq j \leq i\}$ . Order the elements of  $\hat{A}_i \setminus \hat{A}_{i-1}$  in an arbitrary order, and let  $v_j$  denote the  $j$ -th element in this order. Consider some  $1 \leq j \leq |\hat{A}_i \setminus \hat{A}_{i-1}|$ ,

if  $v_j \notin N_{i-1}$  then the pair of the element  $v_j$  and the set  $\{v_1, v_2, \dots, v_{i-1}\} \cap \mathcal{D}^+(v_j)$  form a possible pair that Algorithm 3 could select on iteration  $i$  since  $v_j \in \mathcal{N} \setminus (N_{i-1} \cup \hat{A}_{i-1}) = \hat{T}_{i-1} \setminus \hat{A}_{i-1}$ . Hence,

$$\begin{aligned} f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) &\geq f(v_j \mid \hat{A}_{i-1} \cup (\{v_1, v_2, \dots, v_{i-1}\} \cap \mathcal{D}^+(v_j))) \\ &\geq f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \dots, v_{i-1}\}) . \end{aligned}$$

On the other hand, if  $v_j \in N_{i-1}$ , then there must be some  $1 \leq h < i$  such that  $v_j = \hat{u}_h$ , and thus,  $v_j \in \hat{T}_h$ . By a similar argument to the one used above we get:

$$\begin{aligned} f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) &\geq f(v_j \mid \hat{A}_{h-1} \cup ((\hat{A}_{i-1} \cup \{v_1, v_2, \dots, v_{i-1}\}) \cap \mathcal{D}^+(v_j))) \\ &\geq f(v_j \mid \hat{A}_{h-1} \cup (\hat{A}_{i-1} \cup \{v_1, v_2, \dots, v_{i-1}\})) \\ &= f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \dots, v_{i-1}\}) . \end{aligned}$$

Adding the inequalities we got for every  $1 \leq j \leq |\hat{A}_i \setminus \hat{A}_{i-1}|$  gives:

$$\begin{aligned} f(\hat{A}_i) - f(\hat{A}_{i-1}) &= \sum_{j=1}^{|\hat{A}_i \setminus \hat{A}_{i-1}|} f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \dots, v_{i-1}\}) \\ &= \sum_{v_j \in \hat{A}_i \setminus (\hat{A}_{i-1} \cup N_{i-1})} f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \dots, v_{i-1}\}) + \sum_{v_j \in \hat{A}_i \cap N_{i-1}} f(v_j \mid \hat{A}_{i-1} \cup \{v_1, v_2, \dots, v_{i-1}\}) \\ &\leq |\hat{A}_i \setminus (\hat{A}_{i-1} \cup N_{i-1})| \cdot f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) + \sum_{u_h \in \hat{A}_i \cap N_{i-1}} f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) . \end{aligned}$$

Observe that  $|\hat{A}_i \setminus (\hat{A}_{i-1} \cup N_{i-1})| \leq d + 1$  and  $\hat{A}_i \cap N_{i-1} \subseteq \hat{A}_i \setminus \hat{A}_{i-1} - \hat{u}_i$ . Combining both observations with the last inequality yields:

$$\begin{aligned} f(\hat{A}_i) &\leq f(\hat{A}_{i-1}) + (d + 1) \cdot f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) + \sum_{u_h \in \hat{A}_i \setminus \hat{A}_{i-1} - \hat{u}_i} f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) \\ &\leq f(\hat{A}_{i-1}) + (d + 1) \cdot \sum_{u_h \in \hat{A}_i \setminus \hat{A}_{i-1}} f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) \leq (d + 1) \cdot \sum_{u_h \in \hat{A}_i} f(\hat{u}_h \mid \hat{A}_{h-1} \cup \mathcal{D}_{\hat{u}_h}) , \end{aligned}$$

where the last inequality holds by the induction hypothesis.  $\square$

**Lemma E.18.** *For every  $0 \leq i \leq \hat{\ell}$ ,  $f(OPT_i) + (d + 1) \cdot L_i \geq f(OPT)$ .*

*Proof.* We prove the lemma by induction on  $i$ . For  $i = 0$  the lemma is trivial since  $f(OPT_0) = f(OPT)$ . Next, assume that the lemma holds for  $i - 1 \geq 0$ , and let us prove it for  $i$ . There are two cases to consider. First, let us consider the case where  $\hat{u}_i$  is removed from  $T$  and  $\hat{A}_i = \hat{A}_{i-1}$ . In this case one potential candidate for  $OPT_i$  is  $OPT_{i-1} - \hat{u}_i$ . If  $\hat{u}_i \notin OPT_{i-1}$ , then we get  $f(OPT_{i-1} - \hat{u}_i) = f(OPT_{i-1})$ . On the other hand, if  $\hat{u}_i \in OPT_{i-1}$  then the pair of the element  $\hat{u}_i$  and the set  $(OPT_{i-1} \setminus \hat{A}_{i-1}) \cap \mathcal{D}^+(\hat{u}_i)$  is a possible pair that Algorithm 3 could select on iteration  $i$ . Hence,

$$\begin{aligned} f(OPT_{i-1} - \hat{u}_i) &= f(OPT_{i-1}) - f(\hat{u}_i \mid \hat{A}_{i-1} \cup (OPT_{i-1} \setminus \hat{A}_{i-1} - \hat{u}_i)) \\ &\geq f(OPT_{i-1}) - f(\hat{u}_i \mid \hat{A}_{i-1} \cup ((OPT_{i-1} \setminus \hat{A}_{i-1}) \cap \mathcal{D}^+(\hat{u}_i))) \\ &\geq f(OPT_{i-1}) - f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) . \end{aligned}$$

Therefore, regardless of the membership of  $\hat{u}_i$  in  $OPT_{i-1}$ , we can lower bound by  $f(OPT_{i-1} - \hat{u}_i)$  by  $f(OPT_{i-1}) - f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i})$ . Using the induction hypothesis, we now get:

$$\begin{aligned} f(OPT_i) + (d+1) \cdot L_i &\geq f(OPT_{i-1} - \hat{u}_i) + (d+1) \cdot L_{i-1} + (d+1) \cdot f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) \\ &\geq f(OPT_{i-1}) + (d+1) \cdot L_{i-1} \geq f(OPT) . \end{aligned}$$

Next, let us consider the case where  $\hat{u}_i$  is kept in  $T$  and  $\hat{A}_i = \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i} + \hat{u}_i$ . In this case, by standard matroid properties, one can obtain a candidate for  $OPT_i$  by starting with  $OPT_{i-1} \cup \mathcal{D}_{\hat{u}_i} + \hat{u}_i$  and removing from it a subset  $\Delta \subseteq OPT_{i-1} \setminus \hat{A}_i$  of up to  $d+1$  elements. Let us denote by  $OPT'_i$  this candidate, *i.e.*,  $OPT'_i = (OPT_{i-1} \cup \mathcal{D}_{\hat{u}_i} + \hat{u}_i) \setminus \Delta$ .

Order the elements of  $\Delta$  in an arbitrary order, and let  $v_j$  denote the  $j$ -th element in this order. Observe that by monotonicity:

$$\begin{aligned} f(OPT_{i-1}) - f(OPT'_i) &\leq f(OPT_{i-1}) - f(OPT_{i-1} \setminus \Delta) = \sum_{j=1}^{|\Delta|} f(v_j \mid OPT_{i-1} \setminus \{v_1, v_2, \dots, v_j\}) \\ &\leq \sum_{j=1}^{|\Delta|} f(v_j \mid A_{i-1} \cup ((OPT_{i-1} \setminus \{v_1, v_2, \dots, v_j\}) \cap \mathcal{D}^+(v_j))) \leq (d+1) \cdot f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}^+(\hat{u}_i)) . \end{aligned}$$

where the last inequality holds since  $v_j \in OPT_{i-1} \setminus \hat{A}_i \subseteq \hat{T}_{i-1}$  for every  $1 \leq j \leq |\Delta|$ , and thus, the pair of the element  $v_j$  and the set  $[(OPT_{i-1} \setminus \{v_1, v_2, \dots, v_j\}) \setminus \hat{A}_{i-1}] \cap \mathcal{D}^+(v_j)$  is a possible pair that Algorithm 3 could select on iteration  $i$ . Using the induction hypothesis, we now get:

$$\begin{aligned} f(OPT_i) + (d+1) \cdot L_i &\geq f(OPT'_i) + (d+1) \cdot L_{i-1} + (d+1) \cdot f(\hat{u}_i \mid \hat{A}_{i-1} \cup \mathcal{D}_{\hat{u}_i}) \\ &\geq f(OPT_{i-1}) + (d+1) \cdot L_{i-1} \geq f(OPT) . \end{aligned} \quad \square$$

**Corollary E.19.** *The parameter  $L$  of the process corresponding to Algorithm 3 can be chosen to be  $f(OPT)/[2(d+1)]$ .*

*Proof.* Observe that any value that always lower bounds  $L_{\hat{\ell}}$  can be used as a value for the parameter  $L$ . Combining Lemmata E.17 and E.18 gives:

$$2(d+1) \cdot L_{\hat{\ell}} \geq f(\hat{A}_{\hat{\ell}}) + [f(OPT) - f(OPT_{\hat{\ell}})] = f(OPT) ,$$

where the equality holds by Observation E.16.  $\square$

Now that we have all the parameters of the process corresponding to Algorithm 3, we can use Corollary E.12 to give a guarantee on  $W_{\ell}$ .

**Lemma E.20.** *When Algorithm 2 applies the second option and  $m^* \leq f(OPT)/[256(d+1)^2]$ , then, with probability at least  $7/8$ ,  $W_{\ell} \geq \frac{f(OPT)}{8(d+2)^2}$ .*

*Proof.* Recall that  $\hat{W}_{\ell}$  is the sum of the accepted values in the process corresponding to Algorithm 3. Hence, by Corollary E.12,

$$\begin{aligned} \Pr \left[ \hat{W}_{\ell} < \frac{f(OPT)}{8(d+2)^2} \right] &\leq \Pr \left[ \hat{W}_{\ell} < \frac{pL}{2} - \frac{pL}{4} \right] \leq \frac{pB(L+2B)}{4(pL/4)^2} = \frac{4B(L+2B)}{pL^2} \\ &= \frac{4 \cdot \frac{f(OPT)}{256(d+1)^2} \cdot \left( \frac{f(OPT)}{2(d+1)} + \frac{2 \cdot f(OPT)}{256(d+1)^2} \right)}{\frac{1}{d+2} \cdot \left( \frac{f(OPT)}{2(d+1)} \right)^2} \leq \frac{4 \cdot \frac{1}{256(d+1)^2} \cdot \frac{1}{d+1}}{\frac{1}{d+2} \cdot \frac{1}{4(d+1)^2}} = \frac{(d+2)}{16(d+1)} \leq \frac{1}{8} . \end{aligned}$$

The lemma now follows since  $W_{\ell}$  and  $\hat{W}_{\ell}$  have the same distribution by Observation E.15.  $\square$

To complete the analysis of Algorithm 2 we also need the following notation and lemma. Given a set  $S \subseteq \mathcal{N}$ , let  $S(p)$  be a random set containing every element  $u \in S$ , independently, with probability  $p$ .

**Lemma E.21.** *For every set  $S \subseteq \mathcal{N}$ ,  $\mathbb{E}[f(S(p))] \geq p^{d+1} \cdot f(S)$ .*

*Proof.* For every element  $u \in S$ , let  $X_u$  be an indicator for the event that  $S \cap \mathcal{D}^+(u) + u \in S(p)$ . Clearly,  $\Pr[X_u = 1] \geq p^{d+1}$ . Also, let  $u_1, u_2, \dots, u_{|S|}$  denote an arbitrary order of the elements of  $S$ . Then,

$$\begin{aligned} \mathbb{E}[f(S(p))] &= \sum_{i=1}^{|S|} \mathbb{E}[f(\{u_i\} \cap S(p) \mid S(p) \cap \{u_1, u_2, \dots, u_{i-1}\})] \\ &\geq \sum_{i=1}^{|S|} \mathbb{E}[X_i \cdot f(u_i \mid \{u_1, u_2, \dots, u_{i-1}\})] \\ &\geq p^{d+1} \cdot \sum_{i=1}^{|S|} f(u_i \mid \{u_1, u_2, \dots, u_{i-1}\}) = p^{d+1} \cdot f(S) , \end{aligned}$$

where the first inequality follows from the definition of  $\mathcal{D}^+(u)$ . □

**Lemma E.22.** *If  $m^* \leq f(OPT)/[256(d+1)^2]$ , then Algorithm 2 is  $O(\beta)$ -competitive.*

*Proof.* Throughout this proof we assume that Algorithm 2 applies the second option. This event happens with probability  $1/2$ , thus, it is enough to prove that Algorithm 2 is  $O(\beta)$ -competitive given this event.

Let  $R = \mathcal{N} \setminus T_\ell$  be the set of elements that are not observed by Algorithm 2. By Lemma E.21:

$$\begin{aligned} \mathbb{E}[f(OPT \cap R)] &= \mathbb{E}\left[f\left(OPT \left(1 - \frac{1}{d+2}\right)\right)\right] \\ &\geq \left(1 - \frac{1}{d+2}\right)^{d+1} \cdot f(OPT) \geq e^{-1} \cdot f(OPT) . \end{aligned}$$

Hence,

$$\mathbb{E}[f(OPT(R))] \geq \mathbb{E}[f(OPT \cap R)] \geq \frac{f(OPT)}{e} \geq \frac{f(OPT)}{4} ,$$

where  $OPT(R)$  is the independent subset of  $R$  maximizing  $f$ . Since  $f(OPT(R))$  is always upper bounded by  $f(OPT)$ , this implies the following claim:

$$\Pr\left[f(OPT(R)) \geq \frac{f(OPT)}{10}\right] \geq \frac{1}{6} .$$

On the other hand, by Lemma E.20,  $W_\ell \geq f(OPT)/[8(d+2)^2]$  with probability at least  $7/8$ . Hence, by the union bound we have:

$$W_\ell \geq \frac{f(OPT)}{8(d+2)^2} \quad \text{and} \quad f(OPT(R)) \geq \frac{f(OPT)}{10}$$

with probability at least  $1/24$ . To complete the proof it is enough to show that  $ALG$  is  $O(\beta)$ -competitive when the last two inequalities hold. This follows from the definition of  $ALG$  when

$\text{opt}_{80(d+2)^2}$  is a valid approximation for  $f(OPT(R))$ , *i.e.*, when we have  $f(OPT(R))/[80(d+2)^2] \leq \text{opt}_{80(d+2)^2} \leq f(OPT(R))$ . Thus, in the rest of the proof we prove these inequalities:

$$\text{opt}_{80(d+2)^2} = \frac{W_\ell}{10} \leq \frac{f(OPT)}{10} \leq f(OPT(R)) \ ,$$

where the first inequality follows from Lemma E.14. On the other hand,

$$f(OPT(R)) \leq f(OPT) \leq 8(d+2)^2 W_\ell = 80(d+2) \cdot \text{opt}_{80(d+2)^2} \ . \quad \square$$

Note that Theorem 6.1 follows immediately from Lemmata C.1 and E.22.